

AD-A168 339 CORROS: A COMPUTER PROGRAM FOR ANALYZING POLARIZATION

AD-A168 339 CORROS: A COMPUTER PROGRAM FOR ANALYZING POLARIZATION 1/1

AD-A168 339 CORROS: A COMPUTER PROGRAM FOR ANALYZING POLARIZATION 1/1

RESISTANCE DATA(U) DEFENCE RESEARCH ESTABLISHMENT

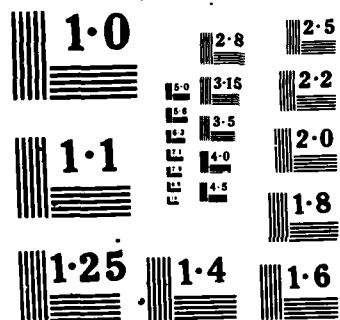
ATLANTIC DARTMOUTH (NOVA SCOTIA) C M HANHAN ET AL

UNCLASSIFIED APR 86 DREA-TC-86/303 F/G 20/3

UNCLASSIFIED APR 86 DREA-TC-86/303 F/G 20/3

UNCLASSIFIED APR 86 DREA-TC-86/303 F/G 20/3

UNCLASSIFIED APR 86 DREA-TC-86/303 F/G 20/3 NL



NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST

AD-A168 339



National Defence
Research and
Development Branch

Défense Nationale
Bureau de Recherche
et Développement

UNLIMITED DISTRIBUTION

3

TECHNICAL COMMUNICATION 86/303

April 1986

CORROS: A COMPUTER PROGRAM
FOR ANALYZING
POLARIZATION RESISTANCE DATA

C.M. Hanham - P.J. Gallagher

DTIC
ELECTE
JUN 04 1986
S D

DTIC FILE COPY

Defence
Research
Establishment
Atlantic



Centre de
Recherches pour la
Défense
Atlantique

Canada

DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

UNLIMITED DISTRIBUTION



National Defence
Research and
Development Branch

Défense Nationale
Bureau de Recherche
et Développement

**CORROS: A COMPUTER PROGRAM
FOR ANALYZING
POLARIZATION RESISTANCE DATA**

C.M. Hanham - P.J. Gallagher

April 1986

Approved by R.S. Hollingshead A/H/Dockyard Laboratory Section

DISTRIBUTION APPROVED BY

A/D/TD

TECHNICAL COMMUNICATION 86/303

**Defence
Research
Establishment
Atlantic**



**Centre de
Recherches pour la
Défense
Atlantique**

Canada

ABSTRACT

A program, CORROS, has been written to run on the DEC 20 computer at DREA. It provides a means for analyzing polarization resistance data from potentiodynamic polarization experiments, in order to determine corrosion current densities. CORROS accepts data files which are transferred from an EG&G PARC Model 350 Corrosion Measurement System to the DEC 20 computer with another computer program, <Staal>C11A20¹. A nonlinear least squares curve fitting technique is used to fit a curve, which satisfies the Stern-Geary equation, to the experimental data. The corrosion potential ϕ_{corr} , the anodic and cathodic Tafel constants b_a and b_c , the polarization resistance R_p , and the corrosion current density i_{corr} , along with their estimated errors, and the relative RMS error of the fitted data, are determined and stored in a file LIST. Another file PLOTS, which is accepted by <Staal>SAPLTF² for graphic presentation of the data, is also output.

SOMMAIRE

Le programme CORROS a été écrit pour être exploité sur l'ordinateur DEC 20 du CRDA. Ce programme permet d'analyser les données sur la résistance à la polarisation obtenues lors d'expériences de polarisation potentiodynamiques, pour déterminer ensuite les densités de courant de corrosion. Le CORROS accepte des fichiers de données qui sont transférés d'un système de mesure de la corrosion EG&G PARC, modèle 350, à l'ordinateur DEC 20 à l'aide d'un autre infoprogramme, le <Staal>C11A20¹. Une technique non linéaire de lissage par la méthode des moindres carrés est utilisée pour ajuster une courbe, qui satisfait l'équation de Stern-Geary, aux données expérimentales. Le potentiel de corrosion ϕ_{corr} , les constantes de Tafel anodique et cathodique b_a et b_c , la résistance à la polarisation R_p et la densité de courant de corrosion i_{corr} , ainsi que les erreurs estimées correspondantes et l'erreur quadratique moyenne des données ajustées, sont déterminés et mis en mémoire dans le fichier LIST. Un autre fichier, le PLOTS, accepté par le programme <Staal>SAPLTF² pour la représentation graphique des données, est aussi produit.

TABLE OF CONTENTS

	PAGE
ABSTRACT	ii
NOTATION	iv
1. INTRODUCTION	1
2. A DESCRIPTION OF CORROS	1
3. RUNNING CORROS	8
4. CORROS VERSUS TAFEL PLOT ANALYSIS	13
5. CONCLUDING REMARKS	13
TABLES	14
FIGURES	15
<u>APPENDIX</u>	<u>PAGE</u>
A. CORROS	18
B. <Staal>C11A20 ¹ For Data Transfer Between EG&G PARC Model 350 and DEC 20 COMPUTER	36
C. Data File (T05F04.DAT)	38
D. Subroutines SPLINE ³⁰ , SEVAL ³⁰ , DECOMP ³⁴ and SOLVE ³⁴	41
E. Simplified CORROS Run	49
F. Detailed CORROS Run	52
G. PLOTS	61
H. LIST	66
I. Donahue's ³⁵ Data	67
J. Simplified CORROS Analysis of Donahue's ³⁵ Data	68
K. PLOTS From CORROS Analysis of Donahue's ³⁵ Data	73
L. LIST From CORROS Analysis of Donahue's ³⁵ Data	75
REFERENCES	76

Accession For	
NTIS	<input checked="" type="checkbox"/>
CRA&I	<input type="checkbox"/>
DTIC	<input type="checkbox"/>
TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	



NOTATION

a	substitution variable
b_a	anodic Tafel constant
b_a	anodic Tafel constant
b_c	cathodic Tafel constant
b_c	cathodic Tafel constant
$B(I)$	cubic interpolating spline coefficient
$C(I)$	cubic interpolating spline coefficient
$D(I)$	cubic interpolating spline coefficient
E_{corr}	corrosion potential
$ECORR$	corrosion potential
EF	final potential
EI	initial potential
$f(x)$	a function
g_i	gradient at i
i	current density
i_{corr}	corrosion current density
I_{corr}	corrosion current density
J_i	Jacobian at i
J_i^{-1}	inverse Jacobian at i
MV/SEC	scan rate (mV/sec)
N	number of data points
oP	$\Delta\phi$ or $\phi - \phi_{corr}$

NOTATION (continued)

r_i	root of $f(x)$ or approximation to the solution
RMS	root mean square
R_p	polarization resistance
Rp	polarization resistance
$S(X)$	cubic interpolating spline
u	substitution variable
v	substitution variable
w	substitution variable
$X(I)$	data point abscissa
$Y(I)$	data point ordinate
α	correction multiplier of b_a
β	correction multiplier of b_c
γ	correction multiplier of i_{corr}
Δr	correction applied to r_i to generate r_{i+1}
$\Delta\phi$	$\phi - \phi_{\text{corr}}$
ϵ	correction multiplier of ϕ_{corr}
ϕ	potential
ϕ_{corr}	corrosion potential

1. INTRODUCTION

Corrosion rates can be determined from weight loss measurements³ or, more rapidly, by less direct electrochemical techniques. These latter techniques are based on a classic paper which correlates the use of anodic and cathodic polarization data and mixed electrode reactions to the steady state (corrosion) potential and the steady state (corrosion) current density⁴. The use of electrochemical techniques for determining corrosion rates and conducting corrosion studies has been rigorously debated⁵, rejected^{6,7}, and praised^{8,9}.

Electrochemical techniques are characterized by the potential range over which the data are measured and by the method of data analysis. Polarization resistance measurements usually cover a small potential range of 50 mV around the corrosion potential¹⁰. The linear polarization method was initially used for the analysis of polarization resistance data in order to determine the slope of a supposed linear portion of the polarization curve^{11,12}. The value of this slope has been called the polarization resistance (R_p) and is inversely proportional to both the corrosion current density and the corrosion rate. This method has more recently been strongly criticized¹³⁻¹⁶ and has resulted in the development of alternative analysis techniques.

One substitutional method which was introduced allowed simultaneous calculation of the polarization resistance, the Tafel slopes, the corrosion potential and the corrosion current density by graphic analysis¹⁷ and later by computer aided methods^{18,19}. A variety of computer programs have been written using linear¹⁹ and nonlinear²⁰ least squares curve fitting techniques, a simplified three point method^{21,22}, and a four point method²³. Programs designed for use in industrial corrosion monitoring systems and which therefore do not require sophisticated computer analysis have also been developed²⁴. A program was written for the analysis of curves which deviate substantially from the family of curves allowed by the Stern-Geary equation because of concentration polarization²⁵. Another was written specifically for microcomputers²⁶ while yet another was based on a visual superposition of empirical and calculated curves through an interactive computer graphics procedure^{27,28}. Each of these programs has unique approaches, advantages and pitfalls. Very few unbiased comparisons of the different computer programs have taken place. It has been demonstrated, however, that the least squares method of analysis is more accurate than the point methods of calculation²⁹.

The computer program CORROS was developed at DREA to run on the DEC 20 computer in order to specifically analyze polarization resistance data obtained through the EG&G PARC Model 350 Corrosion Measurement System. The EG&G PARC Model 350 Corrosion Measurement System has its own software and is capable of determining the polarization resistance R_p from polarization resistance data but does so by the linear polarization method. CORROS, on the other hand, is capable of simultaneously determining the corrosion potential, the Tafel constants, the polarization resistance and the corrosion current density from polarization resistance data obtained from the mixed potential region around the corrosion potential. The method is based on a nonlinear least squares curve fit of the data to the Stern-Geary equation. The output includes the corrosion parameters, their standard deviations, and the relative root mean square error of the fitted data.

2. A DESCRIPTION OF CORROS

CORROS, as listed in Appendix A, is written in the Pascal computer language and can be easily divided into three sections: dataheader, datagenerator and datafit.

Dataheader is designed to read and recognize certain features in a data file. The data is transferred from the EG&G PARC Model 350 Corrosion Measurement System to the DREA DEC 20 computer through the use of <Staal>C11A20¹, as listed in Appendix B. A data file, as listed in Appendix C, is made up of two parts. The

first part contains information about the experiment, such as the sample number, the date, the tape and file numbers where the data is stored, the surface area, the potential range of the data, the potential scan rate, the initially determined corrosion potential, a value for the polarization resistance calculated by the EG&G PARC Model 350 Corrosion Measurement System, and the final determination of the corrosion potential. The second part of a data file is the actual experimental results: pairs of potential/current density measurements. Dataheader recognizes the end of the first part of a data file and the beginning of the second part.

The second section of CORROS, datagenerator, then takes over. It reads the data from a file, converts the potential measurements from volts to millivolts and the current measurements from nanoamps per square centimeter to milliamps per square centimeter, and then passes a cubic spline through the data to allow access to points between the real data points.

For a typical polarization resistance experiment with a potential range of 50 millivolts ($\phi_{\text{corr}} \pm 25 \text{ mV}$), there are 99 data points. Each data point is a potential value in volts and a current density value in nanoamps per square centimeter. The potential value of each data point increases by 0.5 millivolts, although the EG&G PARC Model 350 Corrosion Measurement System does not report the points in this manner. It instead rounds off the potential values to the nearest millivolt or 0.001 volts. Therefore, it gives, what appears at first glance to be, two pairs of data points increasing in potential value by 1 millivolt.

Datagenerator interprets the data file correctly so that each data point increases in potential value by 0.5 millivolts. It compares the potential value of each data point with the potential value of the data point read previously. If the two potentials are equal, it adds a correction of 0.0005 volts to the potential value of the data point read most recently. For example, it would interpret the following data points,

-0.335	-1.769E2		-0.3350	-1.769E2
-0.335	-1.544E2		-0.3345	-1.544E2
-0.334	-1.387E2		-0.3340	-1.387E2
-0.334	-1.202E2	as	-0.3335	-1.202E2
-0.333	-1.084E2		-0.3330	-1.084E2
-0.333	-1.006E2		-0.3325	-1.006E2
-0.332	-9.363E1		-0.3320	-9.363E1

Datagenerator then stores the data in arrays named "potential" and "current" with units of millivolts and milliamps per square centimeter. The ascension of the data is checked to ensure that there are no mistakes which occasionally appear while the data is being transferred from the EG&G PARC Model 350 Corrosion Measurement System to the DEC 20 computer.

The data is then interpolated with a cubic spline through an external Fortran subroutine, SPLINE³⁰. This subroutine is listed in Appendix D. The purpose of interpolation in this program is to have a fast algorithm to calculate current density values for potential values not in the table of data. The program requires interpolated values of potential and current density between the real data points at a later part of the program.

The subroutine, SPLINE, uses a technique called piecewise polynomial interpolation³¹. For a set of data points or knots, $X(N)$ and $Y(N)$, where N is the number of data points, X are the abscissas of the data points, and Y are the ordinates of the data points, the coefficients $B(I)$, $C(I)$, and $D(I)$, where $I=1,2,3,\dots,N$, are computed for a cubic interpolating spline

$$S(X) = Y(I) + B(I)[X - X(I)] + C(I)[X - X(I)]^2 + D(I)[X - X(I)]^3 \quad (1)$$

for

$$X(l) \leq X \leq X(l+1) \quad (2)$$

The third section of CORROS, datafit, first determines the corrosion potential and then screens the data to a symmetric range around this potential. The data range around the initial value of the corrosion potential, determined by the EG&G PARC 350 Corrosion Measurement System, is 25 millivolts on either side. The corrosion potential value actually shifts though, during the polarization of the sample. It is consistently as much as 7 millivolts cathodic to the initially determined value of the corrosion potential. The final data set is therefore lopsided with respect to the corrosion potential. The cathodic range is typically 18 millivolts, while the anodic range is typically 32 millivolts.

The algorithm in the program CORROS is written to handle only symmetric sets of data; i.e., equal ranges of data on either side of the corrosion potential. Only a portion of the data set is therefore analyzed, typically 18 millivolts on either side of the corrosion potential. This means that up to 30% of the data from an experiment is ignored.

The program uses two external Fortran subroutines, EVAL and DEVAL (from SEVAL³⁰), also listed in Appendix D, to find where the rate of change of potential with current on the spline is a minimum. The current changes, at this point, from travelling in a cathodic direction to an anodic direction. EVAL is used to evaluate the spline and DEVAL is used to determine the spline's derivative at a given ordinate value.

The program continues with datafit by determining an initial approximation for the Tafel constants b_a and b_c , the corrosion current density i_{corr} and the polarization resistance R_p , and then by improving this approximation to arrive at the best solution.

CORROS determines its initial approximations of the desired parameters through the simplification of the Stern-Geary equation

$$i = \frac{1}{\ln 10 R_p} \left(\frac{1}{1/b_a + 1/b_c} \right) \left[\exp \frac{\ln 10 (\phi - \phi_{corr})}{b_a} - \exp \frac{-\ln 10 (\phi - \phi_{corr})}{b_c} \right] \quad (3)$$

with several substitutions.

$$\Delta \phi = \phi - \phi_{corr} \quad (4)$$

$$u = \frac{\ln 10}{b_a} \quad (5)$$

$$v = \frac{\ln 10}{b_c} \quad (6)$$

$$u = w + a \quad (7)$$

and

$$v = w - a \quad (8)$$

Then,

$$i = \frac{\exp(\Delta\phi u) - \exp(-\Delta\phi v)}{R_p(u+v)} \quad (9)$$

which is identical to

$$i = \frac{\exp(\Delta\phi a) \sinh(\Delta\phi w)}{R_p w} \quad (10)$$

Also,

$$i_{corr} = \frac{1}{\ln 10 R_p} \left(\frac{1}{1/b_a + 1/b_c} \right) = \frac{1}{R_p(u+v)} = \frac{1}{2 R_p w} \quad (11)$$

Then, combining equations (10) and (11) gives

$$i = 2 i_{corr} \exp(\Delta\phi a) \sinh(\Delta\phi w) \quad (12)$$

The properties of symmetry and asymmetry are then used on equation (12) to give

$$\left[\frac{i(\Delta\phi)}{-i(-\Delta\phi)} \right]^{1/2} = \exp(\Delta\phi a) \quad (13)$$

which contains only the unknown value of a .

The determination of a at various values of $\Delta\phi$ should give an indication of the value of $\Delta\phi$ at which the Tafel constants become stable. As indicated above, the Tafel constants are a function of the value of a , as a result of substitutions (5), (6), (7), and (8).

The program is written to search for the minimum value of $\Delta\phi$ at which a becomes stable, starting at 5 millivolts. The value of a does not normally stabilize until $\Delta\phi$ is typically 8 millivolts. If a does not approach a constant, the data are suspect.

The properties of symmetry and asymmetry are again used on equation (12) to give

$$\left[i(\Delta\phi) \cdot -i(-\Delta\phi) \right]^{1/2} = \frac{\sinh(\Delta\phi w)}{R_p w} \quad (14)$$

which contains only the parameters R_p and w . Also,

$$\left[i(\Delta\phi) - i(-\Delta\phi) \right]^{1/2} = 2 i_{\text{corr}} \sinh(\Delta\phi w) \quad (15)$$

which contains only the parameters i_{corr} and w .

In order to separate out the w value, the previously calculated spline is used to evaluate $1/2\Delta\phi$ and $3/2\Delta\phi$. This, along with the identity

$$\frac{\sinh(3/2 x)}{\sinh(x)} = 2 \cosh(1/2 x) - \frac{1}{2 \cosh(1/2 x)} \quad (16)$$

is used to generate

$$\left[\frac{i(3/2 \Delta\phi) - i(-3/2 \Delta\phi)}{i(\Delta\phi) - i(-\Delta\phi)} \right]^{1/2} = 2 \cosh(\Delta\phi w/2) - \frac{1}{2 \cosh(\Delta\phi w/2)} \quad (17)$$

where

$$\sinh(\Delta\phi w) = R_p w \left[i(\Delta\phi) - i(-\Delta\phi) \right]^{1/2} \quad (18)$$

Equation (17) is a quadratic in the variable

$$2 \cosh(\Delta\phi w/2) \quad (19)$$

and the most positive root is taken to calculate values of w .

The program calculates values of w for the data above the minimum value of $\Delta\phi$ at which the a values become constant. The scatter of the w values also gives an indication of the quality of the data. The average value of the w values found is used in equations (14) and (15).

Values of R_p are generated as a function of $\Delta\phi$ using equation (14). The average values of a , w and R_p are then used to generate initial approximations of b_a , b_c , and i_{corr} where

$$b_a = \frac{\ln 10}{w + a} \quad (20)$$

$$b_c = \frac{\ln 10}{w - a} \quad (21)$$

and

$$i_{\text{corr}} = \frac{1}{2 R_p w} \quad (22)$$

These approximations are then used as the initial parameters for a nonlinear least squares fitting routine which calculates the gradient and the Jacobian matrix to the sum of squared errors, generates a correction to each parameter, and returns improved values. Nonlinear optimization³² by least squares is based on the formulation of the sum of the squared errors as some function. The gradient of this function (simply the derivative in one dimension) is then set equal to zero.

When the error function is a linear combination of the constants sought (ϕ_{corr} , i_{corr} , b_a , b_c and R_p), setting the gradient equal to zero results in a set of linear equations to be solved. When the error function is a nonlinear equation (such as for the algorithm in CORROS), the set of equations to be solved are also nonlinear. An iterative solution is therefore needed.

Newton's method³³ is one means of arriving at this solution. In this method

$$r_{n+1} = r_n - \frac{f(r_n)}{f'(r_n)} \quad (23)$$

where r_i represent approximations to the root of $f(x)=0$. Each new iterate r_{n+1} is found as the unique zero of the tangent line to the curve $y=f(x)$ at r_n .

Newton's method can be expanded into more than one dimension. The problem, then, is to find the zero of the gradient. In more than one dimension, the gradient is a vector and its derivative becomes a matrix, called the Jacobian. The solution is also a vector.

The iteration can be symbolized as

$$\vec{r}_{n+1} = \vec{r}_n - J_{r_n}^{-1} \cdot \vec{g}_{r_n} \quad (24)$$

where r_i represents approximations to the solution, g_i represents the gradient at i , and J_i^{-1} represents the inverse Jacobian evaluated at i .

Simplification gives

$$J_{r_n} \cdot \Delta \vec{r} = -\vec{g}_{r_n} \quad (25)$$

where

$$\Delta \vec{r} = \vec{r}_{n+1} - \vec{r}_n \quad (26)$$

In other words, Δr is the correction that must be applied to r_n to generate r_{n+1} .

The problem is reduced to generating the gradient and the Jacobian and choosing the variables with which

to work. The following familiar equation is used.

$$i = i_{\text{corr}} \left[\exp \frac{\ln 10 \Delta \phi}{b_a} - \exp \frac{-\ln 10 \Delta \phi}{b_c} \right] \quad (27)$$

where

$$\Delta \phi = \phi - \phi_{\text{corr}} \quad (28)$$

This equation contains ϕ_{corr} , b_a , b_c and i_{corr} . R_p is found from

$$\left(\frac{\partial i}{\partial \phi} \right)_{\phi_{\text{corr}}} = \ln 10 i_{\text{corr}} \left[\frac{1}{b_a} + \frac{1}{b_c} \right] = \frac{1}{R_p} \quad (29)$$

Since there are orders of magnitude difference between the actual parameters, ϕ_{corr} , b_a , b_c , and i_{corr} , this routine accepts these values as constants and creates its own set of variables, epsilon (ϵ), alpha (α), beta (β), and gamma (γ). These new variables are put into the equation for i as multipliers of ϕ_{corr} , b_a , b_c , and i_{corr} respectively. This results in

$$i = \gamma i_{\text{corr}} \left[\exp \frac{\ln 10 (\phi - \epsilon \phi_{\text{corr}})}{\alpha b_a} - \exp \frac{-\ln 10 (\phi - \epsilon \phi_{\text{corr}})}{\beta b_c} \right] \quad (30)$$

They are chosen this way so that each of their initial values will be 1 and any size effect that would have existed if ϕ_{corr} , b_a , b_c , and i_{corr} were used directly is negated. After each successful iteration, the constants ϕ_{corr} , b_a , b_c , and i_{corr} are corrected by multiplication by their corresponding new variable and the new variables each assume a value of 1 for the start of the next iteration. This procedure is continued until ϵ , α , β , and γ remain constant at a value of 1 or the routine reaches its maximum allowed number of iterations.

The gradient and Jacobian are calculated, during each iteration, in terms of ϵ , α , β , and γ . Since these correction multipliers remain essentially at a value of 1 throughout, their values do not figure greatly in the formulae.

The gradient is calculated as the sum of the partial derivatives for each correction multiplier over the data in the range being studied. The Jacobian is approximated by divided differences and is also symmetrized by averaging off diagonal terms.

Two external Fortran subroutines, DECOMP and SOLVE³⁴, which are listed in Appendix D, are used to solve the matrix equation for the correction vector. Together they perform an LU decomposition on a matrix and return the condition number of the matrix. The coefficient matrix is passed to DECOMP, and SOLVE uses the LU form along with the solution vector to find a solution. The correction vector is restricted by two conditions:

- i) ϵ , α , β , and γ are not allowed to change by more than 50% per iteration, and
- ii) the root mean square error at the corrected estimate must be less than the root mean square error at the start.

This protects the routine from running away from or overshooting an error.

The desired constants are then output with their estimated error and the predicted data are tabulated with the actual data. The software can be easily modified so that the data is passed to another file which is suitable for graphic presentation, or to a file which just contains results.

3. RUNNING CORROS

There are two versions of a CORROS run. An example of the simplified version is in Appendix E and the detailed version is in Appendix F. The difference between the two versions is the amount of output to the terminal. During a detailed run, the program outputs the gradient, the Jacobian and its construction, the correction vector, and its treatment at each iteration. During a simplified run, the program operates the same routines but suppresses the output. The results and the information supplied to PLOTS and LIST are identical.

The program is initiated by `ex@corrode` which results in the loading of the program CORROS and the subroutines SPLINE³⁰, SEVAL³⁰, DECOMP³⁴ and SOLVE³⁴, which are found in CORSUB.FOR.

@ex@corrode

PASCAL: CORROS

FORTTRAN: CORSUB

SPLINE

SEVAL

DECOMP

SOLVE

LINK: Loading

[LNKXCT CORROS execution]

Execution starts with

INPUT :

after which, the user responds with a carriage return, then

OUTPUT :

after which, the user again responds with a carriage return, and then

PLOTS :

The user now has a choice of either naming the output file, suitable for graphic presentation, PLOTS by responding with a carriage return or giving this file another name such as P05F04.DAT for example. A listing of a PLOTS type file is found in Appendix G. The next output is

DATA :

The user must now give the name of the data file, such as T05F04.DAT, that is to be analyzed. After the next output

LIST :

the user again has a choice of either naming the output file, which contains a summary of results, LIST by responding with a carriage return or giving this file another name such as L05.DAT for example. A listing of a LIST type file is found in Appendix H.

The program next asks the user if they would like to read the introduction.

Would you like to read the introduction ? yes = 1

If the user wishes to do so, they respond with a 1.

1

A program to find the corrosion potential E_{corr} , the anodic and cathodic Tafel constants b_a and b_c , the polarization resistance R_p , and the corrosion current density I_{corr} , from the mixed potential region around E_{corr} . The data is read from polarization resistance data from the EG&G PARC Model 350 Corrosion Measurement System by another program, <Staal>C11A20¹. The program can be run in two modes: detailed and simple. The only difference being the amount of output to the terminal. In detailed mode the program outputs the gradient, the Jacobian and its construction, the correction vector and its treatment at each iteration. The simple mode operates the same routines but suppresses output.

If not, the response is anything except 1. The program then asks the user if a detailed run is required.

do you want a detailed run ? yes = 1

Again 1 means yes and anything else except 1 means no.

After the program reports the information which is in the data header at the beginning of the data file,

*SAMPLE 4 DATE 28.10 TAPE 2 FILE 1 AREA 1.142E1 EI -0.276 EF -0.227
MV/SEC 0.167 ECORR -0.252
RESULTS RP 5.257E3
ECORR -0.256 *** end of tape header ****

and the number of data points in the data file,

99 data points read in

it asks the user if they would like to see a tabulation of the data.

do you want to see the data tabulated? yes = 1

1 again means yes and anything else means no.

The program reports the initial guess for ϕ_{corr} ,

estimated E_{corr} is -255.11

the size of one half of the symmetric range around ϕ_{corr} ,

range around E_{corr} reduced to 20.9

and the number of data points in the symmetric range which are being used in the analysis.

84 points in range

The program next reports the value of a at each 0.5 mV interval of $\Delta\phi$, starting at $\Delta\phi = 5.0$ and going to the end of the symmetric range of data.

<i>at oP = 5.0</i>	<i>a = -0.000173</i>	<i>at oP = 5.5</i>	<i>a = 0.001862</i>
<i>at oP = 6.0</i>	<i>a = 0.002409</i>	<i>at oP = 6.5</i>	<i>a = 0.003409</i>
<i>.</i>	<i>.</i>	<i>.</i>	<i>.</i>
<i>.</i>	<i>.</i>	<i>.</i>	<i>.</i>
<i>.</i>	<i>.</i>	<i>.</i>	<i>.</i>
<i>.</i>	<i>.</i>	<i>.</i>	<i>.</i>
<i>at oP = 19.0</i>	<i>a = 0.003753</i>	<i>at oP = 19.5</i>	<i>a = 0.002325</i>
<i>at oP = 20.0</i>	<i>a = 0.001458</i>	<i>at oP = 20.5</i>	<i>a = -0.000052</i>

After reporting the average value of a , the program asks the user for the value of $\Delta\phi$ at which the user wants the program to continue with the analysis.

average a is 0.006881

enter starting oP: minimum oP where a becomes reasonably constant

The user normally chooses $\Delta\phi = 5$ but by scanning the list of a values, may also choose the minimum value of $\Delta\phi$ at which the value of a becomes reasonably constant. This choice can sometimes solve nonconvergence problems with certain data files.

The program then reports the average value of w ,

average w is 0.057436

an improved reestimate of the value of a ,

re-estimate of a is 0.0069

an average value of R_p ,

average R_p is 5217.

an estimate of I_{corr}

estimated Icorr is 0.00166878

a reestimate of the value of w,

re-estimate of w is 0.05757037

the initial estimates of ϕ_{corr} , b_a , b_c , i_{corr} and R_p ,

*initial Ecorr,ba,bc,Icorr and Rp are
-255.11 35.73 45.43 0.001665 5217.*

and the initial relative root mean square error of the fit.

initial relative RMS error is 0.082915

If the initial values are acceptable, the user responds with a 1.

enter 1 if acceptable

If they are not acceptable, the user responds with anything else except a 1 and is then asked to supply initial estimates for b_a , b_c and R_p .

*2
enter ba bc Rp * in mvolts/decade and ohms *
36 45 5200
relative RMS error = 0.079201
enter 1 if acceptable
1*

The program then asks the user to respond with a 1 if they wish the program to continue and the iterations to begin, or to respond with a 99 if they wish the program to stop.

enter 1 to continue : 99 to quit

After each iteration, if the run is detailed, the program reports the gradient, the Jacobian matrix and the correction vector.

gradient at each parameter + del
5.559737E-04 -2.110495E-05 2.477400E-05 -6.371798E-06
5.498207E-04 -2.090320E-05 2.471899E-05 -6.459422E-06
5.500678E-04 -2.091288E-05 2.472174E-05 -6.454742E-06
5.500761E-04 -2.092529E-05 2.471082E-05 -6.436885E-06

gradient: dEcorr,dba,dbc,dIcorr
5.500043E-04 -2.091514E-05 2.471674E-05 -6.449062E-06

Jacobian before symmetry check
5.969370E-01 -1.836815E-02 6.341725E-03 7.173367E-03

```
-1.898122E-02  1.193507E-03  2.252136E-04  -1.015496E-03
5.726088E-03  2.246452E-04  4.996309E-04  -5.922402E-04
7.726374E-03  -1.036022E-03  -5.680192E-04  1.217643E-03
```

Jacobian matrix

```
5.969370E-01  -1.867469E-02  6.033906E-03  7.449870E-03
-1.867469E-02  1.193507E-03  2.249294E-04  -1.025759E-03
6.033906E-03  2.249294E-04  4.996309E-04  -5.801297E-04
7.449870E-03  -1.025759E-03  -5.801297E-04  1.217643E-03
```

initial correction vector

```
5.176613E-03  3.579467E-01  1.037736E-01  3.246053E-01
```

error overshoot: reduced correction by half

The program then reports improved estimates of ϕ_{corr} , b_a , b_c , i_{corr} and R_p , and the reduced relative root mean square error of the fit for both detailed and simplified runs. The user may stop the iterations at any time by responding with a 99 or may continue by responding with a 1.

```
Ecorr  ba  bc  Icorr  Rp
-255.77  42.12  47.78  0.001935  5024.
relative RMS error now = 0.073427
enter 1 to continue : 99 to quit
1
```

The program reports when there is no further improvement available, the user responds with a 99, and the program reports the final relative root mean square error of the fit, the final estimates of ϕ_{corr} , b_a , b_c , i_{corr} and R_p , and the error in each estimated parameter.

no further improvement available

```
Ecorr  ba  bc  Icorr  Rp
-256.18  34.67  33.74  0.001368  5430.
relative RMS error now = 0.060409
enter 1 to continue : 99 to quit
99
relative RMS error now = 0.060409
Ecorr = -256.18 millivolts
ba = 34.67 +/- 4.640 millivolts
bc = 33.74 +/- 2.949 millivolts
Rp = 5430. +/- 460.8 ohms
Icorr = 0.001368 +/- 0.00011417 millamps /cm2
```

The program finally asks the user for the tape number of the data file and the file number of the data file. Referring to the example, the user would respond with 5 and 4 respectively.

```
TAPE number ?
5
FILE number ?
4
```

A summary of the results of the analysis is found in LIST or under a file name that the user has chosen. This file is appended so it may contain summaries of previous analysis results. The data and the fitted curve to the data may be graphically presented together by running <Staal>SAPLTF² and by giving this program the name of the file which contains the analysis suitable for graphic presentation; either PLOTS or a name chosen by the user. The results of the analysis of data file T05F04.DAT are shown in Figure 1.

4. CORROS VERSUS TAFEL PLOT ANALYSIS

CORROS is also capable of analyzing polarization data over a range larger than that of polarization resistance data. Donahue³⁵ presented an illustration of the analysis of electrochemical rate data for activation controlled reactions, using a Tafel plot technique. For the purpose of a comparison, the illustration was also analyzed with CORROS. Donahue's data is listed in Appendix I. The analysis of the data, using a simplified run of CORROS, is listed in Appendix J. The output PLOTS file and the output LIST file are listed in Appendices K and L respectively. Donahue's Tafel plot analysis is illustrated in Figure 2 while the CORROS analysis, showing the data and the fitted curve, is presented in Figure 3.

The results of the two analysis techniques are significantly close as is shown in Table I.

5. CONCLUDING REMARKS

The computer program CORROS provides a means for the analysis of polarization resistance data from electrochemical corrosion experiments, for the purpose of simultaneously determining corrosion current densities, Tafel slopes, polarization resistance values and corrosion potentials. The results of the analysis are summarised in one output file and are readily available for graphic presentation in another output file. CORROS may be easily used by persons with no detailed knowledge of corrosion science and/or no experience with computers.

TABLE I

COMPARISON OF THE TAFEL PLOT ANALYSIS AND THE CORROS ANALYSIS
OF DONAHUE'S ILLUSTRATION

	<u>ϕ_{corr} (mV)</u>	<u>b_a (mV)</u>	<u>b_c (mV)</u>	<u>R_p (kΩ)</u>	<u>i_{corr} (mA/cm²)</u>
<u>Tafel Plot</u>	200.	40.	120.		1.0
<u>CORROS</u>	200.	40.57 \pm 0.71	126.70 \pm 13.15	0.01 \pm 0.00	1.07 \pm 0.03

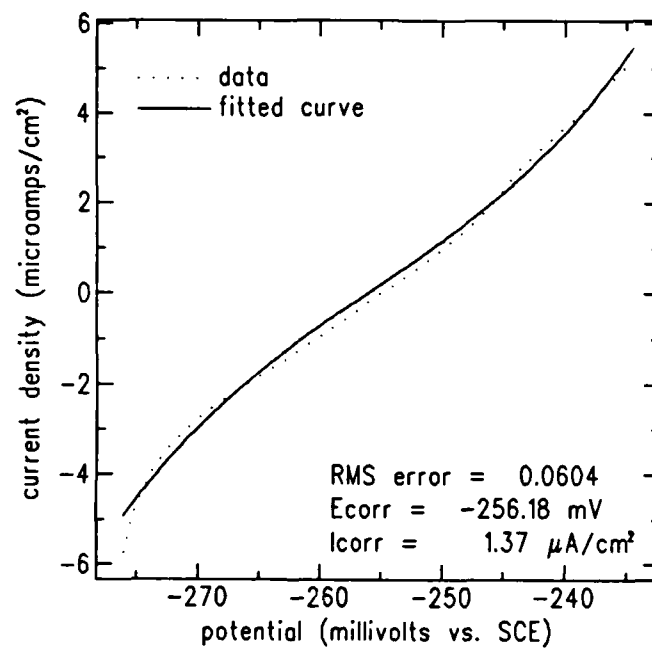


Figure 1 <Staal>SAPLTF² Presentation of the CORROS Analysis of T05F04.DAT

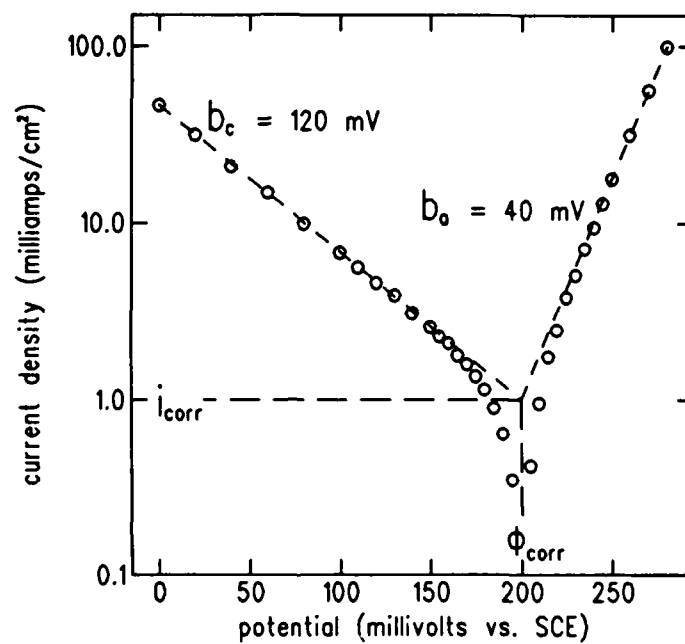


Figure 2 Donahue's³⁵ Tafel Plot

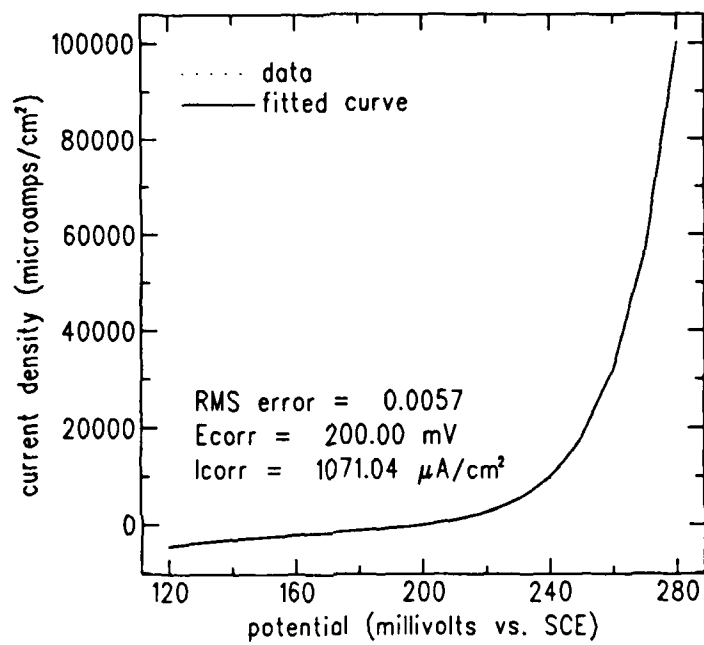


Figure 3 CORROS Analysis of Donahue's³⁵ Illustration

APPENDIX A

CORROS

PROGRAM CorrosionAnalysis(INPUT:/,OUTPUT,PLOTS,DATA,LIST);

```
{ A program to find the corrosion potential Ecorr, the anodic and cathodic tafel constants ba and bc, the }
{ polarization resistance Rp and the corrosion current density Icorr, from the mixed potential region }
{ around Ecorr. The data is read from polarization resistance data from an EG&G PARC Model 350 }
{ Corrosion Measurement System by another program, <Staal>C11A201. The program's basic routine }
{ is a nonlinear least squares fit of the data. Output includes the corrosion parameters, their estimated }
{ errors, and the relative RMS error of the fitted data. These parameters can be found in a file, LIST.. }
{ The real data and the fitted data are also output to another file, PLOTS., and a plot can be output by }
{ running <STAAL>SAPLTF2. }
```

CONST

```
numdata = 200;
order = 4;
```

TYPE

```
index = 1..numdata;
sindex = 1..order;
svector = ARRAY[sindex] of real;
isvector = ARRAY[sindex] of integer;
vector = ARRAY[index] of real;
matrix = ARRAY[sindex,sindex] of real;
```

VAR

DATA,PLOTS,LIST	: text;
gradstore,Jac	: matrix;
gradient,mgradient,work	: svector;
potential,current,voltdata,currdatabb,CC,DD	: vector;
ipvt	: isvector;
i,j,m,n,numofpts,count,ans,asn	: integer;
Rp,Ecorr,Icorr,ba,bc,cond	: real;
detailed	: boolean;

PROCEDURE dataheader;

CONST

```
size = 6;
blank = ' ';
```

TYPE

```
wordtype = ARRAY[1..size] of char;
testtype = PACKED ARRAY[1..size] of char;
```

```

VAR
    ch          : char;
    i           : integer;
    testword,keyword : testtype;
    seive       : wordtype;

PROCEDURE skipblanks;

BEGIN
    REPEAT read(DATA,ch) UNTIL (ch <> blank);
END;

PROCEDURE initseive;

BEGIN
    skipblanks;
    seive[1] := ch;
    FOR i := 2 TO size DO
        BEGIN
            read(DATA,ch);
            seive[i] := ch;
        END;
    END;

PROCEDURE lookfor(keyword:testtype);

BEGIN
    initseive;
    REPEAT
        write(seive[1]);
        read(DATA,ch);
        FOR i := 1 TO size-1 DO
            seive[i] := seive[i+1];
        seive[size] := ch;
        pack(seive,1,testword);
        UNTIL (testword = keyword);
        writeln;
        FOR i := 1 TO size DO
            write(seive[i]);
        END;

BEGIN { dataheader }

    writeln;
    (' Would you like to read the introduction ? yes = 1');
    readln; read(ans);
    IF ans = 1
    THEN

```

BEGIN

```
writeln(' A program to find the corrosion potential Ecorr, the anodic and cathodic tafel constants ba and bc, ');
writeln(' the polarization resistance Rp, and the corrosion current density Icorr, from the mixed potential ');
writeln(' region around Ecorr. The data is read from polarization resistance data from an EG&G PARC ');
writeln(' Model 350 Corrosion Measurement System by another program, <STAAL>C11A20. The ');
writeln(' program can be run in two modes: detailed and simple. The only difference being the amount of ');
writeln(' output to the terminal. In detailed mode the program outputs the gradient, the Jacobian and its ');
writeln(' construction, the correction vector and its treatment at each iteration. The simple mode operates ');
writeln(' the same routines but suppresses output. ');
```

END;

```
writeln;
writeln(' do you want a detailed run ? yes = 1 ');
readln; read(ans);
IF ans = 1 THEN detailed := TRUE
  ELSE detailed := FALSE;
```

```
reset(DATA);
lookfor('RESULT');
lookfor('ECORR ');
read(DATA,ch); write(ch);
WHILE NOT eoln(DATA) DO
  BEGIN
    read(DATA,ch); write(ch);
  END;
  writeln(' *** end of tape header ***');
  writeln;
END; { dataheader }
```

```
{ These routines are FORTRAN subprograms supplied at load time. }
{ All four can be found in CORSUB.FOR. }
}
```

PROCEDURE spline(n : integer; xdata,ydata : vector; VAR b,c,d : vector);

{ fits an interpolating cubic spline to a set of data of strictly increasing ordinate (x) value }

extern FORTRAN;

PROCEDURE seval(n : integer; xval : real; xdata,ydata,b,c,d : vector; VAR yval,dyval : real);

{ evaluates the spline (y) and its deravitive (y') at a given ordinate (x) value }

extern FORTRAN;

PROCEDURE decomp(ndim,n : integer; VAR A : matrix; VAR cond : real; VAR ipvt : isvector;
VAR work : svector);

{ performs LU decomposition of an n x n matrix A and returns the condition number of the matrix }

```

extern FORTRAN;

PROCEDURE solve(ndim,n : integer; VAR A : matrix; VAR B :svector; VAR ipvt : isvector);

    { solves a set of n simultaneous equations in n variables. The coefficient matrix A is passed to      }
    { DECOMP and SOLVE and uses the LU form along with the solution vector B to solve Ax = B          }

extern FORTRAN;

PROCEDURE DataGenerator;

    { takes raw data from a file.DAT and converts poteniials from volts to mVolts and current densities from      }
    { namps to mAmps per sqcm and passes a cubic spline through the data to allow access to points 'between' }
    { data points.                                                                                       }

VAR
    voltage,amperage,vstp : real;

PROCEDURE scale;    { scales volts to mVolts and namps to mamps }

BEGIN
    FOR i := 1 TO numofpts DO
        BEGIN
            potential[i] := potential[i] * 1E3;
            current[i] := current[i] / 1E6;
        END;
    END;

PROCEDURE reorder;    { reorders decreasing voltage data }

VAR    tempvolt,tempamp : real;
        i,nexti         : integer;

BEGIN
    writeln('reordering');
    FOR i := 1 TO (numofpts DIV 2) DO
        BEGIN
            nexti         := numofpts + 1 - i;

            tempvolt      := potential[i];    tempamp      := current[i];
            potential[i]  := potential[nexti]; current[i]   := current[nexti];
            potential[nexti] := tempvolt;      current[nexti] := tempamp;

        END;
    END; { reorder }

PROCEDURE viewdata;

```

```

BEGIN

writeln; writeln('do you want to see the data tabulated? yes = 1');
readln; read(ans);
IF ans = 1 THEN
BEGIN
  writeln('*****');
  writeln;
  writeln(' Potential    Current      Potential    Current');
  writeln;
  FOR j := 1 TO numofpts DIV 2 DO
    writeln(j:2,potential[j]:8:2,current[j]:15:6,
      (j + numofpts DIV 2):8,potential[j+numofpts DIV 2]:8:2,
      current[j+numofpts DIV 2]:16:6 );

    writeln('*****');
  END;
END; { viewdata }

BEGIN { datagenerator }

  i := 1;
  read(DATA,voltage,amperage);
  IF amperage < 0 THEN vstp := 0.0005 ELSE vstp := -0.0005;

  WHILE NOT eof(DATA) DO
    BEGIN
      potential[i] := voltage;
      current[i] := amperage;
      read(DATA,voltage,amperage);

      { this section detects data which have the same voltage because the EG&G PARC Model 350 Corrosion }
      { Measurement System can only pass 3 digits and either adds or subtracts half of the last digit to those }
      { voltages depending on whether the voltages are increasing or not }

      IF voltage = potential[i] THEN voltage := voltage + vstp;

      i := i + 1;

    END; { WHILE }

    numofpts := i - 1;
    scale;
    writeln(numofpts:3,' data points read in ');
    IF potential[1] > potential[4] THEN reorder;
    viewdata;
    spline(numofpts,potential,current,BB,CC,DD);

  END; { datagenerator }

```

```
PROCEDURE screendata(window : real);
```

```
    { screens out data that is more than a 'window' away from Ecorr }
```

```
    BEGIN
```

```
        FOR i := 1 TO numdata DO
```

```
            BEGIN
```

```
                voldtdata[i] := 0; currddata[i] := 0;
```

```
            END;
```

```
    j := 0;
```

```
        FOR i := 1 TO numofpts DO
```

```
            BEGIN
```

```
                IF abs(potential[i] - Ecorr) <= window
```

```
                THEN
```

```
                    BEGIN
```

```
                        j := j + 1;
```

```
                        voldtdata[j] := potential[i];
```

```
                        currddata[j] := current[i];
```

```
                    END;
```

```
            END;
```

```
        count := j;
```

```
        writeln(count:4, ' points in range ');
```

```
    END; { screendata }
```

```
PROCEDURE datafit;
```

```
    { let L = ln(10), Ecorr be the corrosion potential in mvolts and E - Ecorr be the overpotential oP in }
    { millivolts. Let I be the total current density in mamps and Icorr be the corrosion current density in }
    { mamps. Then the equation that governs I wrt P is: 1. I = Icorr * (exp(L * oP/ba) - exp(-L * oP/bc)) }
    { where ba and bc are the anodic and cathodic Tafel constants. Further, the slope of the I vs oP curve at }
    { oP = 0 is a constant @ oP=0, (dI/doP) = 1/Rp, where Rp is the polarization resistance and @ }
    { oP=0, (dI/doP) = Icorr * ln(10) * (1/ba + 1/bc). Thus 2. 1/Icorr = L * Rp * (1/ba + 1/bc). Now }
    { substitute u = L/ba and v = L/bc and 2. into 1. and get }
    { 3. I = (exp(oP * u) - exp(-oP * v))/(Rp * (u + v)). Now substitute u = w + a and v = w - a and get }
    { 4. I = exp(P * a) * sinh(P * w)/(Rp * w). The variables in equation 4 can be partly or wholly }
    { separated. Consider I = I(oP) : i) I(oP)/I(-oP) = -exp(2 * oP * a) contains only a, }
    { ii) sqrt(I(oP)*I(-oP)) = sinh(oP * w)/(Rp * w) only Rp and w, or substituting Icorr into (ii), }
    { iii) sqrt(I(oP)*I(-oP)) = 2 * Icorr * sinh(oP * w). One variable which has not been dealt with yet, Ecorr, }
    { is hidden in P or oP and since these appear as arguments to the exponential, it is very necessary to }
    { accurately determine Ecorr. The initial approximation to Ecorr is the potential E at which the current }
    { density I(E) = 0. This E is found by solving for the root of spln(E) = 0, where spln is the cubic spline }
    { which interpolates the data and this is done using Newton's method, x1 = x0 - y/y'. }

```

```
VAR
```

```
    V, oP, oPincr, minoP, w0, sum1, sum2, sum3, sum4, yval, dyval, temp1, temp2, L, kp, delta,
    w, sumw, sdevw, estEcorr, range, a, suma, sdeva, den, z, IoP, Iplus, Iminus, Ip3half, Im3half, oPrange,
    rms, newrms, epsilon, alpha, beta, gamma, sdevIcorr, sdevba, sdevbc, scdevRp : real;
```

```

i,j,num,rej : integer;
FUNCTION eval( V : real) : real;

{ returns, from the data spline, a current value for a given input potential in mvolts }

BEGIN
    seval(numofpts,V,potential,current,BB,CC,DD,yval,dyval);
    eval := yval;
END;

FUNCTION deval( V : real) : real;

{ returns the rate of change of current at a potential }

BEGIN
    seval(numofpts,V,potential,current,BB,CC,DD,yval,dyval);
    deval := dyval;
END;

FUNCTION fitcurrent(V,Ecorr,ba,bc,Icorr : real) : real;

BEGIN
    oP := V - Ecorr;
    IF oP <> 0
    THEN
        fitcurrent := Icorr * (exp(L*oP/ba) - exp(-L*oP/bc))
    ELSE
        fitcurrent := 0.0;
    END;
END;

FUNCTION rmserr(Ecorr,ba,bc,Icorr : real) : real;

VAR sum1,sum2 : real;
    ij : integer;

BEGIN
    sum1 := 0; sum2 := 0; j := 0;

    FOR i := 1 TO count DO
    BEGIN
        oP := voltdata[i] - Ecorr;
        IF ((abs(oP) >= minOP)
            AND (abs(oP) <= oPrange + 1))
        THEN
        BEGIN
            sum1 := sum1 + sqr(currdata[i] - fitcurrent(oP + Ecorr,Ecorr,ba,bc,Icorr));
            sum2 := sum2 + sqr(currdata[i]);
            j := j + 1;
        END;
    END;
END;

```

```

END;
rmserr := sqrt(sum1/sum2);
END;

```

```

PROCEDURE improve(VAR Ecorr,ba,bc,Icorr : real);

```

```

{ This routine takes initial values of Ecorr, ba, bc, and Icorr and returns improved values, least squares }
{ sense, by applying a nonlinear Newton iteration in four variables. The iteration is of the form }
{  $J0 * dX = -G(X)$  where dX is the correction vector, J0 is the Jacobian matrix evaluated at X0 and G0 }
{ is the gradient of the sum of squared errors at X0, the initial parameters. Since there are orders of }
{ magnitude difference between the actual parameters Ecorr, ba, bc, and Icorr this routine accepts these }
{ values as constants and creates its own set of variables epsilon, alpha, beta, and gamma. These new }
{ variables are put into the equation for I as multipliers of Ecorr, ba, bc, and Icorr respectively. They are }
{ chosen this way so that each of their initial values will be 1 (one) and this negates any size effect that }
{ would have existed if Ecorr, ba, bc, and Icorr were used directly. After each successful iteration, the }
{ 'constants' Ecorr, ba, bc, and Icorr are corrected by multiplication by their corresponding new variable }
{ and the new variables each assume a value of 1 (one) for the start of the next iteration. The procedure }
{ is continued until epsilon, alpha, beta, and gamma remain constant at a value of 1 (one) or the routine }
{ iterates past its maximum allowed runs (user set, default of 50). }

```

```

VAR
temp1, temp2, predcurr, error, pepsilon, palpha, pbeta, pgamma : real;
i,j,m,n : integer;

```

```

PROCEDURE grad(eps,alp,beta,gam : real);

```

```

VAR maxgrad : real;

```

```

BEGIN

```

```

{ 'p' is used to represent differentiation }

```

```

pepsilon := 0; pgamma := 0; palpha := 0; pbeta := 0;

```

```

FOR i := 1 TO count DO

```

```

BEGIN

```

```

oP := voltdata[i] - eps*Ecorr;

```

```

IF ((abs(oP) >= minOP)

```

```

AND (abs(oP) <= oPrange + 1))

```

```

THEN

```

```

BEGIN

```

```

predcurr := fitcurrent(voltdata[i],eps*Ecorr,alp*ba,beta*bc,gam*Icorr);

```

```

error := currdata[i] - predcurr;

```

```

temp1 := exp( L * oP/ba)/ba;

```

```

temp2 := exp(-L * oP/bc)/bc;

```

```

pepsilon := pepsilon + (2 * error * Icorr * L * Ecorr * (temp1 + temp2));

```

```

    pgamma := pgamma + (-2 * error * predcurr);

    palpha  := palpha + (2 * error * Icorr * L * oP * temp1);

    pbeta   := pbeta + (2 * error * Icorr * L * oP * temp2);

    END;
END;

gradient[1] := pepsilon;
gradient[2] := palpha;
gradient[3] := pbeta;
gradient[4] := pgamma;

END; { grad }

PROCEDURE jacobian(Ecorr,ba,bc,Icorr : real);

{ The Jacobian matrix is made up of the derivatives of the gradient in the form: Jac[i,j] = dgrad[i]/dX[i]. }
{ This routine approximates the derivatives by a divided difference and further assures that the matrix is }
{ exactly symmetric, accounting for some roundoff. }

CONST del = 1E-5;

BEGIN

    grad(1+del,1,1,1);
    FOR m := 1 TO 4 DO
        gradstore[1,m] := gradient[m];

        grad(1,1+del,1,1);
        FOR m := 1 TO 4 DO
            gradstore[2,m] := gradient[m];

            grad(1,1,1+del,1);
            FOR m := 1 TO 4 DO
                gradstore[3,m] := gradient[m];

                grad(1,1,1,1+del);
                FOR m := 1 TO 4 DO
                    gradstore[4,m] := gradient[m];

            writeln;

            { Writes out the gradient at the 4 points around current estimate, for use in finding Jacobian. }

        IF detailed THEN
            BEGIN

```

```

        writeln(' gradient at each parameter + del');
    FOR m := 1 TO 4 DO
        BEGIN
            FOR n := 1 TO 4 DO
                write(gradstore[m,n]);
                writeln;
            END;
            writeln;
        END;

    grad(1,1,1,1);
    IF detailed THEN
        BEGIN
            writeln(' gradient: dEcorr,dba,dbc,dIcorr');
            FOR m:= 1 TO 4 DO
                write(gradient[m]);
                writeln;writeln;
            END;

    FOR m := 1 TO 4 DO
        FOR n := 1 TO 4 DO
            Jac[m,n] := (gradstore[n,m] - gradient[m])/del;

{ Writes out the Jacobian before forced symmetry. }

    IF detailed THEN
        BEGIN
            writeln(' Jacobian before symmetry check ');
            FOR m := 1 TO 4 DO
                BEGIN
                    FOR n := 1 TO 4 DO
                        write(Jac[m,n]);
                        writeln;
                    END;
                END;
        END;

        writeln;
        FOR m := 1 TO 4 DO
            FOR n := 1 TO 4 DO
                BEGIN
                    IF m <> n
                        THEN
                            IF m < n
                                THEN Jac[m,n] := (Jac[m,n] + Jac[n,m])/2;
                            END;
                END;
            FOR m := 1 TO 4 DO
                FOR n := 1 TO 4 DO
                    IF m > n
                        THEN Jac[m,n] := Jac[n,m];

```

```

IF detailed THEN
  BEGIN
    writeln(' Jacobian matrix ');
    FOR m := 1 TO 4 DO
      BEGIN
        FOR n := 1 TO 4 DO
          write(Jac[m,n]);
          writeln;
        END;
        writeln;
      END;

    FOR m := 1 TO 4 DO
      mgradient[m] := -gradient[m];

    END; { jacobian }

  BEGIN { improve }

    jacobian(Ecorr,ba,bc,Icorr);
    decomp(order,order,Jac,cond,ipvt,work);
    IF (cond = cond + 1)
      THEN
        BEGIN
          IF detailed THEN
            writeln(' jacobian is singular to working precision ');
            FOR n := 1 TO 4 DO
              mgradient[n] := 20 * mgradient[n];
            IF detailed THEN
              writeln(' negative gradient followed to a new point ');
            END
          ELSE
            BEGIN
              solve(order,order,Jac,mgradient,ipvt);
            END;
          END;

        IF detailed
          THEN
            BEGIN
              writeln(' initial correction vector ');
              FOR n := 1 TO 4 DO
                write(mgradient[n]);
                writeln;
              END;
            END;

          WHILE ((abs(mgradient[1]) > 0.5)
            OR (abs(mgradient[2]) > 0.5)
            OR (abs(mgradient[3]) > 0.5)
            OR (abs(mgradient[4]) > 0.5)) DO

```

```

BEGIN
  FOR i := 1 TO 4 DO
    mgradiant[i] := mgradiant[i]/2;
    IF detailed THEN
      write('50% parameter change exceeded:');
    IF detailed THEN
      writeln(' reduced correction by half ');
  END; { while }

  newrms := rmserr(Ecorr*(1+mgradient[1]),ba*(1+mgradient[2]),bc*(1+mgradient[3]),
                  Icorr*(1+mgradient[4]));

  j := 0;
  WHILE ((newrms > rms) AND (j < 25)) DO
    BEGIN
      FOR i := 1 TO 4 DO
        mgradiant[i] := mgradiant[i]/2;
      IF detailed THEN
        writeln(' error overshoot: reduced correction by half ');
      newrms := rmserr(Ecorr*(1+mgradient[1]),ba*(1+mgradient[2]),bc*(1+mgradient[3]),
                      Icorr*(1+mgradient[4]));

      j := j + 1;
      IF j = 25
      THEN writeln(' no further improvement available ');

    END; { while }

    epsilon := 1 + mgradiant[1];
    alpha := 1 + mgradiant[2];
    beta := 1 + mgradiant[3];
    gamma := 1 + mgradiant[4];

    IF ((epsilon = 1) AND (alpha = 1) AND (beta = 1) AND (gamma = 1))
    THEN
      BEGIN
        writeln(' no further improvement available ');
      END;

    Ecorr := Ecorr * epsilon;
    ba := ba * alpha;
    bc := bc * beta;
    Icorr := Icorr * gamma;

    IF ((ba <= 0) OR (bc <= 0))
    THEN
      BEGIN
        writeln(' negative tafel constants reached ');

```

```

        END;
Rp := ((ba * bc)/(ba + bc))/(L * Icorr);
writeln;
writeln(' Ecorr   ba      bc      Icorr   Rp');
writeln(Ecorr:8:2,ba:10:2,bc:10:2,Icorr:12:6,Rp:12:0);
END;

```

PROCEDURE stats;

```

BEGIN
j := 0; m := 0; n := 0;
sdevIcorr := 0; sdevba := 0; sdevbc := 0; sdevRp := 0;
FOR i := 1 TO count DO
    BEGIN
        oP := voltdata[i] - Ecorr;
        IF ((abs(oP) >= minOP) AND (abs(oP) <= oPrange))
        THEN
            BEGIN
                j := j + 1;
                IoP := currdata[i];
                temp1 := exp( L * oP/ba);
                temp2 := exp(-L * oP/bc);
                sdevIcorr := sdevIcorr + sqr(Icorr - IoP/(temp1 - temp2));
                sdevRp := sdevRp + sqr(Rp - ((temp1 - temp2)/(L * (1/ba + 1/bc) * IoP)));
            END
        ELSE
            BEGIN
                n := n + 1;
                sdevbc := sdevbc + sqr(bc + L * oP/ln(temp1 - IoP/Icorr));
            END;
        END;
    END;
    sdevIcorr := sqrt(sdevIcorr/(j - 1));
    sdevRp := sqrt(sdevRp/(j - 1));
    sdevba := sqrt(sdevba/(m - 1));
    sdevbc := sqrt(sdevbc/(n - 1));
END; { stats }

BEGIN { datafit }

L := ln(10);

{ This is a Newton's iteration loop for Ecorr. The initial guess is the rest potential Erest. }

```

```

estEcorr := (potential[1] + potential[numofpts])/2;
REPEAT
    Ecorr := estEcorr;
    estEcorr := Ecorr - (eval(Ecorr)/deval(Ecorr));
UNTIL (abs((estEcorr - Ecorr)/Ecorr) <= 1E-6);

    Ecorr := estEcorr;
    writeln(' estimated Ecorr is ',Ecorr:6:2);

IF abs(potential[1] - Ecorr) <= abs(potential[numofpts] - Ecorr)
    THEN oPrange := abs(potential[1] - Ecorr)
    ELSE oPrange := abs(potential[numofpts] - Ecorr);
    writeln(' range around Ecorr reduced to ',oPrange:4:1);
    range := oPrange;
    screendata(range);

sum1 := 0; sum2 := 0; sum3 := 0;
oP := 5; i := 0;
REPEAT
    i := i + 1;
    Iplus := eval(oP + Ecorr);
    Iminus := eval(-oP + Ecorr);
    a := ln(abs(Iplus/Iminus))/(2 * oP);
    IF odd(i)
        THEN
            BEGIN
                write(' at oP = ',oP:5:1);
                write(' a = ',a:8:6);
            END
        ELSE
            BEGIN
                write(' at oP = ',oP:5:1);
                writeln(' a = ',a:8:6);
            END;
    sum1 := sum1 + a;
    oP := oP + 0.5;
    UNTIL oP > oPrange;
    num := i;
    a := sum1/num;
    writeln(' average a is ',a:8:6);

REPEAT
    write('enter starting oP:');
    writeln(' minimum oP where a becomes reasonably constant');
    readln; read(minoP); ans := 99;
    IF minoP <= 2
        THEN
            BEGIN

```

```

        writeln(' warning : do not start at less than 2');
        ans := 98;
    END
ELSE IF minoP >= oPrange
THEN
    BEGIN
        writeln(' warning : data only good to ',oPrange:4:1);
        ans := 98;
    END;
UNTIL (ans <> 98);

i := 0; sum1 := 0; sum2 := 0;
oP := minoP;
REPEAT
    Iplus      := eval(oP + Ecorr);
    Ip3half    := eval(1.5*oP + Ecorr);
    Iminus     := eval(-oP + Ecorr);
    Im3half    := eval(-1.5*oP + Ecorr);
    temp1      := sqrt(abs((Ip3half * Im3half)/(Iplus * Iminus)))/2;
    temp1      := (temp1 + sqrt(sqr(temp1) + 1))/2;
    IF temp1 > 1.0
    THEN
        BEGIN
            w := 2 * ln(temp1 + sqrt(sqr(temp1) - 1))/oP;
            i := i + 1;
            sum1 := sum1 + w;
        END
    ELSE
        writeln(' data too scattered at ',oP:4:1,' to give a w ');
    oP := oP + 0.5;
    UNTIL (oP >= 2 * oPrange/3);
    num := i;
    w := sum1/num;
    writeln(' average w is ',w:8:6);

i      := 0;
sum1   := 0;
oP     := minoP;
REPEAT
    Iplus      := eval(oP + Ecorr);
    Iminus     := eval(-oP + Ecorr);
    a          := ln(abs(Iplus/Iminus))/(2 * oP);
    sum2       := sum2 + a;
    Rp         := sinh(w * oP)/(w * (sqrt(abs(Iplus * Iminus))));
    IF detailed THEN
        sum1 := sum1 + Rp;
        i   := i + 1;
        oP  := oP + 0.5;
    UNTIL (oP >= oPrange);

```

```

num := i;
a := sum2/num;
Rp := sum1/num;
writeln('re-estimate of a is ',a:6:4,' average Rp is ',Rp:8:0);
Icorr := 1/(Rp * 2 * w);
writeln(' estimated Icorr is ',Icorr:10:8);

sum1 := 0; i := 0; oP := -oPrange;
REPEAT
  IF (abs(oP) >= minOP)
  THEN
    BEGIN
      i := i + 1;
      IoP := abs(eval(oP + Ecorr));
      temp1 := IoP / (2 * Icorr * exp(a * oP));
      w := (ln(temp1 + sqrt(sqr(temp1) + 1))) / (abs(oP));
      writeln(' at ',oP:4:1,' w = ',w:8:6);
      sum1 := sum1 + w;
    END;
  oP := oP + 0.5;
  UNTIL (oP >= oPrange);
  num := i;
  w := sum1/num;
  writeln('re-estimate of w is ',w:10:8);
  ba := L/(w + a); bc := L/(w - a); Icorr := 1/(2 * w * Rp);
  writeln(' initial Ecorr,ba,bc,Icorr and Rp are ');
  writeln(Ecorr:6:2,ba:10:2,bc:10:2,Icorr:12:6,Rp:12:0);
  rms := rmserr(Ecorr,ba,bc,Icorr);
  writeln(' initial relative RMS error is ',rms:8:6);
  writeln(' enter 1 if acceptable ');
  readln; read(ans);
  WHILE ans <> 1 DO
    BEGIN
      writeln(' enter ba bc Rp * in mvolts/decade and ohms *');
      readln; read(ba,bc,Rp);
      Icorr := ((ba * bc)/(ba + bc))/(L * Rp);
      rms := rmserr(Ecorr,ba,bc,Icorr);
      writeln('relative RMS error = ',rms:8:6);
      writeln(' enter 1 if acceptable ');
      readln; read(ans);
    END;
  REPEAT
    writeln('enter 1 to continue : 99 to quit');
    readln; read(ans);
    IF ans = 1
    THEN improve(Ecorr,ba,bc,Icorr);
    rms := newrms;
    writeln('relative RMS error now = ',rms:8:6);

```

```

UNTIL ans = 99;
stats;
writeln(' Ecorr = ',Ecorr:6:2,' millivolts ');
writeln(' ba   = ',ba:6:2,' +/- ',sdevba:6:3,' millivolts ');
writeln(' bc   = ',bc:6:2,' +/- ',sdevbc:6:3,' millivolts ');
writeln(' Rp   = ',Rp:8:0,' +/- ',sdevRp:6:1,' ohms ');
writeln(' Icorr = ',Icorr:8:6,' +/- ',sdevIcorr:10:8,' millamps ');

writeln;

append(LIST);
writeln(' TAPE number ? ');
readln; read(asn);
writeln(' FILE number ? ');
readln; read(ans);
writeln(LIST);
writeln(LIST,'-----');
writeln(LIST,' File ',ans:3);
writeln(LIST,' relative RMS error = ',rms:8:6);
writeln(LIST,' Ecorr = ',Ecorr:7:2,' millivolts ');
writeln(LIST,' Icorr = ',Icorr*1E3:7:5,' +/- ',sdevIcorr*1E3:7:5,' microamps ');
writeln(LIST,' ba   = ',ba:6:2,' +/- ',sdevba:6:2,' millivolts ');
writeln(LIST,' bc   = ',bc:6:2,' +/- ',sdevbc:6:2,' millivolts ');
writeln(LIST,' Rp   = ',Rp*1E-3:6:2,' +/- ',sdevRp*1E-3:5:2,' kohms ');
writeln(LIST,'-----');

```

```

{ This section copies the data and the predicted data to the file named PLOTS.. which is acceptable for use }
{ with the plotting routine <STAAL>SAPLTF2, a simple curve plotter. }

```

```

rewrite(PLOTS);
writeln(PLOTS,'CSET 1 0');
writeln(PLOTS,'CSET 2 0');
writeln(PLOTS,'LSET 1 6');
writeln(PLOTS,'LSET 2 0');
writeln(PLOTS,'PSET 2. 95. ');
writeln(PLOTS,'DENSIT 0');
writeln(PLOTS,'LABEL 1');
writeln(PLOTS,'potential (millivolts vs. SCE)');
writeln(PLOTS,'LABEL 2');
writeln(PLOTS,'current density (microamps/cm!SUP;2!BAK;)');
writeln(PLOTS,'LABEL -4217');
writeln(PLOTS,'RMS error = ',rms:6:4);
writeln(PLOTS,'LABEL -4211');
writeln(PLOTS,'Ecorr = ',Ecorr:7:2,' mV ');
writeln(PLOTS,'LABEL -4205');
writeln(PLOTS,'Icorr = ',Icorr*1E3:7:2,' !FNT13;m!FNT3;A/cm!SUP;2!BAK; ');
writeln(PLOTS,'LEGEND');
writeln(PLOTS,'data');
writeln(PLOTS,'fitted curve');

```

```

        writeln(PLOTS,'CURVE');
        FOR i := 1 TO count DO
            writeln(PLOTS,voltdata[i],currdata[i] * 1E3);
            writeln(PLOTS,'CURVE');
            FOR i := 1 TO count DO
                writeln(PLOTS,voltdata[i],fitcurrent(voltdata[i],Ecorr,ba,bc,Icorr) * 1E3);
            END;
        END;

BEGIN { MAIN }

    dataheader;

    { Resets the DATA file and reads in and writes the information from the start of the file and leaves the file }
    { open at the start of the actual data. }

    datagenerator;

    { Reads in the raw data which is in volts and namps and stores it in arrays POTENTIAL and CURRENT }
    { in mvolts and mamps. POTENTIAL is checked to ensure it is in ascending order, then an interpolating }
    { cubic spline is passed through the data. }

    datafit

    { Determines constants Ecorr, ba, bc, Icorr and Rp by first making substitutions for the constants then }
    { separating the equation into parts each with one or two constants. Then solves each equation for }
    { approximations to its constant(s). These are then used as the initial parameters to a nonlinear least }
    { squares fitting routine which calculates the gradient and the Jacobian matrix to the sum of squared }
    { errors and generates a correction to each parameter. The iteration continues until the size of the }
    { corrections becomes small enough that no further improvement can be made. The constants are then }
    { output with their estimated error and the predicted data are tabulated with the actual data. }

END. { main }

```

APPENDIX B

<Staal>C11A20¹ For Data Transfer Between EG&G PARC Model 350 and DEC 20 COMPUTER

!<STAAL>C11A20.CMD

4:40pm Monday, 22 February 1982

! This sets up the terminal for PDP 11/34 to DEC20 connection

TER ADM3A

TER WIDTH 255

TER FORMFEED

TER NO RAISE

TER NO INDICATE

REFUSE SYSTEM-MESSAGES

REFUSE LINKS

REFUSE ADVICE

C-----DREA-SWA-STAAL----

PROGRAM C11A20

C

C Written by Philip Staal 2:36pm Thursday, 6 November 1980

C

C PURPOSE: This program types an asterisk, accepts a file-name, types an asterisk, and accepts lines of ASCII
C characters to be put in the file. The maximum line length is 255 characters. When
C @@@[CR][LF] is received, the file is closed and the program stops.

C

DOUBLE PRECISION FILE
INTEGER BUF(255)

C

C ACCEPT FILE NAME, OPEN FILE, AND SEND *

C

TYPE 10010
ACCEPT 10020,FILE
OPEN(UNIT=1,DEVICE='DSK',FILE=FILE)
TYPE 10010

C

C ACCEPT LINES, DETERMINE LENGTH, AND WRITE TO FILE

C

10 ACCEPT 10030,(BUF(J),J=1,255)

DO 20 I=1,255

LAST=256-I

20 IF(BUF(LAST).NE.32) GO TO 30

30 IF(BUF(LAST).EQ.64) GO TO 50

40 WRITE(1,10030) (BUF(K),K=1,LAST)

GO TO 10

C

C IF LAST CHARACTER OF LINE IS @, CHECK FOR TWO PREVIOUS @'S

C
50 IF(BUF(LAST-1).NE.64.OR.BUF(LAST-2).NE.64) GO TO 40
CLOSE(UNIT=1)
STOP

C
10010 FORMAT(' *\$')
10020 FORMAT(A10)
10030 FORMAT(255R1)
END

C-----DREA-SWA-STAAAL----

APPENDIX C

Data File (T05F04.DAT)

This is the format of a typical data file as it is transferred from the EG&G PARC Model 350 Corrosion Measurement System to the DREA DEC 20 computer. The name of this file is T05F04.DAT. This name also refers to the fourth file (F) on tape (T) number five with respect to the tapes used in the EG&G PARC Model 355 Tape Storage Device.

This file is actually double spaced when it is received from the 350 but is shown here as single spaced, to save space.

@type t05f04.dat

SAMPLE 4
DATE 28.10
TAPE 2
FILE 1
AREA 1.142E1
EI -0.276
EF -0.227
MV/SEC 0.167
ECORR -0.252

RESULTS

RP 5.257E3
ECORR -0.256

-0.276 -5.727E3
-0.276 -5.063E3
-0.275 -4.662E3
-0.275 -4.388E3
-0.274 -4.046E3
-0.274 -3.811E3
-0.273 -3.616E3
-0.273 -3.440E3
-0.272 -3.284E3
-0.272 -3.147E3
-0.271 -3.020E3
-0.271 -2.883E3
-0.270 -2.766E3
-0.270 -2.658E3
-0.269 -2.551E3
-0.269 -2.453E3
-0.268 -2.355E3
-0.268 -2.267E3

-0.267 -2.169E3
-0.267 -2.081E3
-0.266 -1.993E3
-0.266 -1.905E3
-0.265 -1.827E3
-0.265 -1.739E3
-0.264 -1.651E3
-0.264 -1.573E3
-0.263 -1.485E3
-0.263 -1.397E3
-0.262 -1.299E3
-0.262 -1.221E3
-0.261 -1.133E3
-0.261 -1.036E3
-0.260 -9.490E2
-0.260 -8.601E2
-0.259 -7.633E2
-0.259 -6.666E2
-0.258 -5.669E2
-0.258 -4.662E2
-0.257 -3.675E2
-0.257 -2.619E2
-0.256 -1.612E2
-0.256 -6.998E1
-0.255 2.023E1
-0.255 1.094E2
-0.254 2.003E2
-0.254 2.922E2
-0.253 3.743E2
-0.253 4.623E2
-0.252 5.629E2
-0.252 6.685E2
-0.251 7.780E2
-0.251 8.806E2
-0.250 9.920E2
-0.250 1.104E3
-0.249 1.212E3
-0.249 1.319E3
-0.248 1.436E3
-0.248 1.554E3
-0.247 1.671E3
-0.247 1.818E3
-0.246 1.974E3
-0.246 2.130E3
-0.245 2.296E3
-0.245 2.472E3
-0.244 2.629E3
-0.244 2.795E3
-0.243 2.932E3

-0.243 3.069E3
-0.242 3.205E3
-0.242 3.323E3
-0.241 3.440E3
-0.241 3.567E3
-0.240 3.694E3
-0.240 3.821E3
-0.239 3.948E3
-0.239 4.085E3
-0.238 4.232E3
-0.238 4.378E3
-0.237 4.525E3
-0.237 4.652E3
-0.236 4.779E3
-0.236 4.906E3
-0.235 5.053E3
-0.235 5.209E3
-0.234 5.356E3
-0.234 5.502E3
-0.233 5.649E3
-0.233 5.805E3
-0.232 5.962E3
-0.232 6.099E3
-0.231 6.226E3
-0.231 6.353E3
-0.230 6.499E3
-0.230 6.626E3
-0.229 6.753E3
-0.229 6.900E3
-0.228 7.027E3
-0.228 7.144E3
-0.227 7.340E3

APPENDIX D

Subroutines SPLINE³⁰, SEVAL³⁰, DECOMP³⁴ and SOLVE³⁴

```
SUBROUTINE SPLINE (N, X, Y, B, C, D)
  INTEGER N
  REAL X(N), Y(N), B(N), C(N), D(N)
  C
  C THE COEFFICIENTS B(I), C(I), AND D(I), I=1,2,...,N ARE COMPUTED
  C FOR A CUBIC INTERPOLATING SPLINE
  C
  C  $S(X) = Y(I) + B(I)*(X-X(I)) + C(I)*(X-X(I))^{**2} + D(I)*(X-X(I))^{**3}$ 
  C
  C FOR X(I) .LE. X .LE. X(I+1)
  C
  C INPUT..
  C
  C N = THE NUMBER OF DATA POINTS OR KNOTS (N.GE.2)
  C X = THE ABSCISSAS OF THE KNOTS IN STRICTLY INCREASING ORDER
  C Y = THE ORDINATES OF THE KNOTS
  C
  C OUTPUT..
  C
  C B, C, D = ARRAYS OF SPLINE COEFFICIENTS AS DEFINED ABOVE.
  C
  C USING P TO DENOTE DIFFERENTIATION,
  C
  C  $Y(I) = S(X(I))$ 
  C  $B(I) = SP(X(I))$ 
  C  $C(I) = SPP(X(I))/2$ 
  C  $D(I) = SPPP(X(I))/6$  (DERIVATIVE FROM THE RIGHT)
  C
  C THE ACCOMPANYING SUBROUTINE SEVAL CAN BE USED
  C TO EVALUATE THE SPLINE AND ITS DERIVATIVE.
  C
  C
  C INTEGER NM1, IB, I
  C REAL T
  C
  C NM1 = N-1
  C IF ( N .LT. 2 ) RETURN
  C IF ( N .LT. 3 ) GO TO 50
  C
  C SET UP TRIDIAGONAL SYSTEM
  C
  C B = DIAGONAL, D = OFFDIAGONAL, C = RIGHT HAND SIDE.
  C
```

```

D(1) = X(2) - X(1)
C(2) = (Y(2) - Y(1))/D(1)
DO 10 I = 2, NM1
  D(I) = X(I+1) - X(I)
  B(I) = 2.*(D(I-1) + D(I))
  C(I+1) = (Y(I+1) - Y(I))/D(I)
  C(I) = C(I+1) - C(I)
10 CONTINUE
C
C END CONDITIONS. THIRD DERIVATIVES AT X(1) AND X(N)
C OBTAINED FROM DIVIDED DIFFERENCES
C
  B(1) = -D(1)
  B(N) = -D(N-1)
  C(1) = 0.
  C(N) = 0.
  IF ( N.EQ. 3 ) GO TO 15
  C(1) = C(3)/(X(4)-X(2)) - C(2)/(X(3)-X(1))
  C(N) = C(N-1)/(X(N)-X(N-2)) - C(N-2)/(X(N-1)-X(N-3))
  C(1) = C(1)*D(1)**2/(X(4)-X(1))
  C(N) = -C(N)*D(N-1)**2/(X(N)-X(N-3))
C
C FORWARD ELIMINATION
C
15 DO 20 I = 2, N
  T = D(I-1)/B(I-1)
  B(I) = B(I) - T*D(I-1)
  C(I) = C(I) - T*C(I-1)
20 CONTINUE
C
C BACK SUBSTITUTION
C
  C(N) = C(N)/B(N)
  DO 30 IB = 1, NM1
    I = N-IB
    C(I) = (C(I) - D(I)*C(I+1))/B(I)
30 CONTINUE
C
C COMPUTE POLYNOMIAL COEFFICIENTS
C
  B(N) = (Y(N) - Y(NM1))/D(NM1) + D(NM1)*(C(NM1) + 2.*C(N))
  DO 40 I = 1, NM1
    B(I) = (Y(I+1) - Y(I))/D(I) - D(I)*(C(I+1) + 2.*C(I))
    D(I) = (C(I+1) - C(I))/D(I)
    C(I) = 3.*C(I)
40 CONTINUE
  C(N) = 3.*C(N)
  D(N) = D(N-1)
  RETURN

```

```

C
50 B(1) = (Y(2)-Y(1))/(X(2)-X(1))
   C(1) = 0.
   D(1) = 0.
   RETURN
   END
C
C
SUBROUTINE SEVAL(N, U, X, Y, B, C, D, VAL, DVAL)
INTEGER N
REAL U, X(N), Y(N), B(N), C(N), D(N), VAL, DVAL
C
C THIS SUBROUTINE EVALUATES THE CUBIC SPLINE FUNCTION
C AND ITS DERIVATIVE AND RETURNS VAL AND DVAL
C
C VAL = Y(I) + B(I)*(U-X(I)) + C(I)*(U-X(I))**2 + D(I)*(U-X(I))**3
C DVAL = B(I) + 2*C(I)*(U-X(I)) + 3*D(I)*(U-X(I))**2
C WHERE X(I) .LT. U .LT. X(I+1), USING HORNER'S RULE
C
C IF U .LT. X(1) THEN I = 1 IS USED.
C IF U .GE. X(N) THEN I = N IS USED.
C
C INPUT..
C
C N = THE NUMBER OF DATA POINTS
C U = THE ABSCISSA AT WHICH THE SPLINE IS TO BE EVALUATED
C X,Y = THE ARRAYS OF DATA ABSCISSAS AND ORDINATES
C B,C,D = ARRAYS OF SPLINE COEFFICIENTS COMPUTED BY SPLINE
C
C IF U IS NOT IN THE SAME INTERVAL AS THE PREVIOUS CALL, THEN A
C BINARY SEARCH IS PERFORMED TO DETERMINE THE PROPER INTERVAL.
C
INTEGER I, J, K
REAL DX
DATA I/1/
IF ( I .GE. N ) I = 1
IF ( U .LT. X(I) ) GO TO 10
IF ( U .LE. X(I+1) ) GO TO 30
C
C BINARY SEARCH
C
10 I = 1
   J = N+1
20 K = (I+J)/2
   IF ( U .LT. X(K) ) J = K
   IF ( U .GE. X(K) ) I = K
   IF ( J .GT. I+1 ) GO TO 20
C
C EVALUATE SPLINE AND DERIVATIVE

```

```

C
30 DX = U - X(I)
  VAL = Y(I) + DX*(B(I) + DX*(C(I) + DX*D(I)))
  DVAL = B(I) + DX*(2*C(I) + DX*(3*D(I)))
  RETURN
END
SUBROUTINE DECOMP(NDIM,N,A,COND,IPVT,WORK)
C
  INTEGER NDIM,N
  REAL A(NDIM,N),COND,WORK(N)
  INTEGER IPVT(N)
C
C  DECOMPOSES A REAL MATRIX BY GAUSSIAN ELIMINATION
C  AND ESTIMATES THE CONDITION OF THE MATRIX.
C
C  USE SOLVE TO COMPUTE SOLUTIONS TO LINEAR SYSTEMS.
C
C  INPUT..
C
C  NDIM = DECLARED ROW DIMENSION OF THE ARRAY CONTAINING A.
C
C  N = ORDER OF THE MATRIX.
C
C  A = MATRIX TO BE TRIANGULARIZED.
C
C  OUTPUT..
C
C  A CONTAINS AN UPPER TRIANGULAR MATRIX U AND A PERMUTED
C  VERSION OF A LOWER TRIANGULAR MATRIX I-L SO THAT
C  (PERMUTATION MATRIX)*A = L*U .
C
C  COND = AN ESTIMATE OF THE CONDITION OF A .
C  FOR THE LINEAR SYSTEM A*X = B, CHANGES IN A AND B
C  MAY CAUSE CHANGES COND TIMES AS LARGE IN X .
C  IF COND + 1.0 .EQ. COND, A IS SINGULAR TO WORKING PRECISION.
C  COND = 1.0D+32 IF EXACT SINGULARITY IS DETECTED.
C
C  IPVT = THE PIVOT VECTOR.
C  IPVT(K) = THE INDEX OF THE K-TH PIVOT ROW
C  IPVT(N) = (-1)**(NUMBER OF INTERCHANGES)
C
C  WORK SPACE.. THE VECTOR WORK MUST BE DECLARED AND INCLUDED
C  IN THE CALL. ITS INPUT CONTENTS ARE IGNORED.
C  ITS OUTPUT CONTENTS ARE USUALLY UNIMPORTANT.
C
C  THE DETERMINANT OF A CAN BE OBTAINED ON OUTPUT BY
C  DET(A) = IPVT(N) * A(1,1) * A(2,2) * ... * A(N,N).
C
  REAL EK, T, ANORM, YNORM, ZNORM

```

```

C REAL DABS                                !ch
INTEGER NM1, I, J, K, KP1, KB, KM1, M
C
  IPVT(N) = 1
  IF (N .EQ. 1) GO TO 80
  NM1 = N - 1
C
C COMPUTE 1-NORM OF A
C
  ANORM = 0.0D0
  DO 10 J = 1, N
    T = 0.0D0
    DO 5 I = 1, N
      a(i,j)=dble(a(i,j))                !ch
      T = T + ABS(A(I,J))                !ch
5    CONTINUE
    IF (T .GT. ANORM) ANORM = T
10   CONTINUE
C
C GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
  DO 35 K = 1, NM1
    KP1= K+1
C
C FIND PIVOT
C
    M = K
    DO 15 I = KP1, N
      a(i,k)=dble(a(i,k))                !ch
      a(m,k)=dble(a(m,k))                !ch
      IF (ABS(A(I,K)) .GT. ABS(A(M,K))) M = I    !ch
15   CONTINUE
      IPVT(K) = M
      IF (M .NE. K) IPVT(N) = -IPVT(N)
      T = A(M,K)
      A(M,K) = A(K,K)
      A(K,K) = T
C
C SKIP STEP IF PIVOT IS ZERO
C
      IF (T .EQ. 0.0D0) GO TO 35
C
C COMPUTE MULTIPLIERS
C
      DO 20 I = KP1, N
        A(I,K) = -A(I,K)/T
20   CONTINUE
C
C INTERCHANGE AND ELIMINATE BY COLUMNS

```

```

C
DO 30 J = KP1,N
  T = A(M,J)
  A(M,J) = A(K,J)
  A(K,J) = T
  IF (T .EQ. 0.0D0) GO TO 30
DO 25 I = KP1,N
  A(I,J) = A(I,J) + A(I,K)*T
25  CONTINUE
30  CONTINUE
35 CONTINUE
C
C  COND = (1-NORM OF A)*(AN ESTIMATE OF 1-NORM OF A-INVERSE)
C  ESTIMATE OBTAINED BY ONE STEP OF INVERSE ITERATION FOR THE
C  SMALL SINGULAR VECTOR. THIS INVOLVES SOLVING TWO SYSTEMS
C  OF EQUATIONS, (A-TRANSPOSE)*Y = E AND A*Z = Y WHERE E
C  IS A VECTOR OF +1 OR -1 CHOSEN TO CAUSE GROWTH IN Y.
C  ESTIMATE = (1-NORM OF Z)/(1-NORM OF Y)
C
C  SOLVE (A-TRANSPOSE)*Y = E
C
DO 50 K = 1, N
  T = 0.0D0
  IF (K .EQ. 1) GO TO 45
  KM1 = K-1
  DO 40 I = 1, KM1
    T = T + A(I,K)*WORK(I)
40  CONTINUE
45  EK = 1.0D0
  IF (T .LT. 0.0D0) EK = -1.0D0
  IF (A(K,K) .EQ. 0.0D0) GO TO 90
  WORK(K) = -(EK + T)/A(K,K)
50 CONTINUE
DO 60 KB = 1, NM1
  K = N - KB
  T = 0.0D0
  KP1 = K+1
  DO 55 I = KP1, N
    T = T + A(I,K)*WORK(K)
55  CONTINUE
  WORK(K) = T
  M = IPVT(K)
  IF (M .EQ. K) GO TO 60
  T = WORK(M)
  WORK(M) = WORK(K)
  WORK(K) = T
60 CONTINUE
C
  YNORM = 0.0D0

```

```

DO 65 I = 1, N
    work(i)=dble(work(i))          !ch
    YNORM = YNORM + ABS(WORK(I))    !ch
65 CONTINUE
C
C   SOLVE A*Z = Y
C
CALL SOLVE(NDIM, N, A, WORK, IPVT)
C
ZNORM = 0.0D0
DO 70 I = 1, N
    work(i)=dble(work(i))          !ch
    ZNORM = ZNORM + ABS(WORK(I))    !ch
70 CONTINUE
C
C   ESTIMATE CONDITION
C
COND = ANORM*ZNORM/YNORM
IF (COND .LT. 1.0D0) COND = 1.0D0
RETURN
C
C   1-BY-1
C
80 COND = 1.0D0
IF (A(1,1) .NE. 0.0D0) RETURN
C
C   EXACT SINGULARITY
C
90 COND = 1.0D+32
RETURN
END
C
C
SUBROUTINE SOLVE(NDIM, N, A, B, IPVT)
C
INTEGER NDIM, N, IPVT(N)
REAL A(NDIM,N),B(N)
C
C   SOLUTION OF LINEAR SYSTEM, A*X = B .
C   DO NOT USE IF DECOMP HAS DETECTED SINGULARITY.
C
C   INPUT..
C
C   NDIM = DECLARED ROW DIMENSION OF ARRAY CONTAINING A .
C
C   N = ORDER OF MATRIX.
C
C   A = TRIANGULARIZED MATRIX OBTAINED FROM DECOMP .
C

```

```

C  B = RIGHT HAND SIDE VECTOR.
C
C  IPVT = PIVOT VECTOR OBTAINED FROM DECOMP .
C
C  OUTPUT..
C
C  B = SOLUTION VECTOR, X .
C
  INTEGER KB, KM1, NM1, KP1, I, K, M
  REAL T
C
C  FORWARD ELIMINATION
C
  IF (N .EQ. 1) GO TO 50
  NM1 = N-1
  DO 20 K = 1, NM1
    KP1 = K+1
    M = IPVT(K)
    T = B(M)
    B(M) = B(K)
    B(K) = T
    DO 10 I = KP1, N
      B(I) = B(I) + A(I,K)*T
10  CONTINUE
20  CONTINUE
C
C  BACK SUBSTITUTION
C
  DO 40 KB = 1,NM1
    KM1 = N-KB
    K = KM1+1
    B(K) = B(K)/A(K,K)
    T = -B(K)
    DO 30 I = 1, KM1
      B(I) = B(I) + A(I,K)*T
30  CONTINUE
40  CONTINUE
50  B(1) = B(1)/A(1,1)
  RETURN
  END

```

APPENDIX E

Simplified CORROS Run

oP, in the following, is equivalent to $\Delta\phi$ or $\phi - \phi_{\text{corr}}$.

@ex@corrode

PASCAL: CORROS
FORTRAN: CORSUB
SPLINE
SEVAL
DECOMP
SOLVE
LINK: Loading
[LNKXCT CORROS execution]
INPUT :
OUTPUT :
PLOTS :
DATA : t05f04.dat
LIST :

Would you like to read the introduction ? yes = 1
2

do you want a detailed run ? yes = 1
2

SAMPLE 4 DATE 28.10 TAPE 2 FILE 1 AREA 1.142E1 EI -0.276 EF -0.227
MV/SEC 0.167 E CORR -0.252
RESULTS RP 5.257E3
E CORR -0.256 *** end of tape header ***

99 data points read in

do you want to see the data tabulated? yes = 1
2

estimated Ecorr is -255.11
range around Ecorr reduced to 20.9
84 points in range

at oP = 5.0	a = -0.000173	at oP = 5.5	a = 0.001862
at oP = 6.0	a = 0.002409	at oP = 6.5	a = 0.003409
at oP = 7.0	a = 0.004673	at oP = 7.5	a = 0.004996
at oP = 8.0	a = 0.005475	at oP = 8.5	a = 0.006706
at oP = 9.0	a = 0.008307	at oP = 9.5	a = 0.009173
at oP = 10.0	a = 0.010085	at oP = 10.5	a = 0.011198
at oP = 11.0	a = 0.011510	at oP = 11.5	a = 0.011866
at oP = 12.0	a = 0.011721	at oP = 12.5	a = 0.011348

at oP = 13.0	a = 0.011157	at oP = 13.5	a = 0.010626
at oP = 14.0	a = 0.010070	at oP = 14.5	a = 0.009549
at oP = 15.0	a = 0.009089	at oP = 15.5	a = 0.008512
at oP = 16.0	a = 0.007840	at oP = 16.5	a = 0.007387
at oP = 17.0	a = 0.006931	at oP = 17.5	a = 0.006360
at oP = 18.0	a = 0.005709	at oP = 18.5	a = 0.004908
at oP = 19.0	a = 0.003753	at oP = 19.5	a = 0.002325
at oP = 20.0	a = 0.001458	at oP = 20.5	a = -0.000052

average a is 0.006881
enter starting oP: minimum oP where a becomes reasonably constant
5

average w is 0.057436
re-estimate of a is 0.0069 average Rp is 5217.
estimated Icorr is 0.00166878
re-estimate of w is 0.05757037
initial Ecorr,ba,bc,Icorr and Rp are
-255.11 35.73 45.43 0.001665 5217.
initial relative RMS error is 0.082915
enter 1 if acceptable
1

enter 1 to continue : 99 to quit
1

Ecorr	ba	bc	Icorr	Rp	
-255.77	42.12	47.78	0.001935	5024.	

relative RMS error now = 0.073427
enter 1 to continue : 99 to quit
1

Ecorr	ba	bc	Icorr	Rp	
-255.86	37.33	37.39	0.001556	5215.	

relative RMS error now = 0.063821
enter 1 to continue : 99 to quit
1

Ecorr	ba	bc	Icorr	Rp	
-256.20	34.92	33.73	0.001376	5415.	

relative RMS error now = 0.060482
enter 1 to continue : 99 to quit
1

Ecorr	ba	bc	Icorr	Rp	
-256.18	34.66	33.74	0.001367	5430.	

relative RMS error now = 0.060409
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
-256.18 34.67 33.74 0.001368 5430.
relative RMS error now = 0.060409
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
-256.18 34.67 33.74 0.001368 5430.
relative RMS error now = 0.060409
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
-256.18 34.67 33.74 0.001368 5430.
relative RMS error now = 0.060409
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
-256.18 34.67 33.74 0.001368 5430.
relative RMS error now = 0.060409
enter 1 to continue : 99 to quit
1

no further improvement available

Ecorr ba bc Icorr Rp
-256.18 34.67 33.74 0.001368 5430.
relative RMS error now = 0.060409
enter 1 to continue : 99 to quit
99
relative RMS error now = 0.060409
Ecorr = -256.18 millivolts
ba = 34.67 +/- 4.640 millivolts
bc = 33.74 +/- 2.949 millivolts
Rp = 5430. +/- 460.8 ohms
Icorr = 0.001368 +/- 0.00011417 millamps/cm²

TAPE number ?

5

FILE number ?

4

APPENDIX F

Detailed CORROS Run

@start

INPUT :
OUTPUT :
PLOTS :
DATA : t05f04.dat
LIST :

Would you like to read the introduction ? yes = 1

1

A program to find the corrosion potential E_{corr} , the anodic and cathodic Tafel constants b_a and b_c , the polarization resistance R_p and the corrosion current density I_{corr} , from the mixed potential region around E_{corr} . The data is read from polarization resistance data from the EG&G PARC Model 350 Corrosion Measurement System by another program, <Staal>C11A20¹. The program can be run in two modes: detailed and simple. The only difference being the amount of output to the terminal. In detailed mode the program outputs the gradient, the Jacobian and its construction, the correction vector and its treatment at each iteration. The simple mode operates the same routines but suppresses output.

do you want a detailed run ? yes = 1

1

SAMPLE 4 DATE 28.10 TAPE 2 FILE 1 AREA 1.142E1 EI -0.276 EF -0.227
MV/SEC 0.167 ECORR -0.252
RESULTS RP 5.257E3
ECORR -0.256 *** end of tape header ***

99 data points read in

do you want to see the data tabulated? yes = 1

1

	Potential	Current		Potential	Current
1	-276.00	-0.005727	50	-251.50	0.000669
2	-275.50	-0.005063	51	-251.00	0.000778
3	-275.00	-0.004662	52	-250.50	0.000881
4	-274.50	-0.004388	53	-250.00	0.000992
5	-274.00	-0.004046	54	-249.50	0.001104
6	-273.50	-0.003811	55	-249.00	0.001212
7	-273.00	-0.003616	56	-248.50	0.001319
8	-272.50	-0.003440	57	-248.00	0.001436
9	-272.00	-0.003284	58	-247.50	0.001554
10	-271.50	-0.003147	59	-247.00	0.001671

11	-271.00	-0.003020	60	-246.50	0.001818
12	-270.50	-0.002883	61	-246.00	0.001974
13	-270.00	-0.002766	62	-245.50	0.002130
14	-269.50	-0.002658	63	-245.00	0.002296
15	-269.00	-0.002551	64	-244.50	0.002472
16	-268.50	-0.002453	65	-244.00	0.002629
17	-268.00	-0.002355	66	-243.50	0.002795
18	-267.50	-0.002267	67	-243.00	0.002932
19	-267.00	-0.002169	68	-242.50	0.003069
20	-266.50	-0.002081	69	-242.00	0.003205
21	-266.00	-0.001993	70	-241.50	0.003323
22	-265.50	-0.001905	71	-241.00	0.003440
23	-265.00	-0.001827	72	-240.50	0.003567
24	-264.50	-0.001739	73	-240.00	0.003694
25	-264.00	-0.001651	74	-239.50	0.003821
26	-263.50	-0.001573	75	-239.00	0.003948
27	-263.00	-0.001485	76	-238.50	0.004085
28	-262.50	-0.001397	77	-238.00	0.004232
29	-262.00	-0.001299	78	-237.50	0.004378
30	-261.50	-0.001221	79	-237.00	0.004525
31	-261.00	-0.001133	80	-236.50	0.004652
32	-260.50	-0.001036	81	-236.00	0.004779
33	-260.00	-0.000949	82	-235.50	0.004906
34	-259.50	-0.000860	83	-235.00	0.005053
35	-259.00	-0.000763	84	-234.50	0.005209
36	-258.50	-0.000667	85	-234.00	0.005356
37	-258.00	-0.000567	86	-233.50	0.005502
38	-257.50	-0.000466	87	-233.00	0.005649
39	-257.0	-0.000368	88	-232.50	0.005805
40	-256.50	-0.000262	89	-232.00	0.005962
41	-256.00	-0.000161	90	-231.50	0.006099
42	-255.50	-0.000070	91	-231.00	0.006226
43	-255.00	0.000020	92	-230.50	0.006353
44	-254.50	0.000109	93	-230.00	0.006499
45	-254.00	0.000200	94	-229.50	0.006626
46	-253.50	0.000292	95	-229.00	0.006753
47	-253.00	0.000374	96	-228.50	0.006900
48	-252.50	0.000462	97	-228.00	0.007027
49	-252.00	0.000563	98	-227.50	0.007144

estimated Ecorr is -255.11

range around Ecorr reduced to 20.9

84 points in range

at oP = 5.0	a = -0.000173	at oP = 5.5	a = 0.001862
at oP = 6.0	a = 0.002409	at oP = 6.5	a = 0.003409
at oP = 7.0	a = 0.004673	at oP = 7.5	a = 0.004996
at oP = 8.0	a = 0.005475	at oP = 8.5	a = 0.006706
at oP = 9.0	a = 0.008307	at oP = 9.5	a = 0.009173
at oP = 10.0	a = 0.010085	at oP = 10.5	a = 0.011198

at oP = 11.0	a = 0.011510	at oP = 11.5	a = 0.011866
at oP = 12.0	a = 0.011721	at oP = 12.5	a = 0.011348
at oP = 13.0	a = 0.011157	at oP = 13.5	a = 0.010626
at oP = 14.0	a = 0.010070	at oP = 14.5	a = 0.009549
at oP = 15.0	a = 0.009089	at oP = 15.5	a = 0.008512
at oP = 16.0	a = 0.007840	at oP = 16.5	a = 0.007387
at oP = 17.0	a = 0.006931	at oP = 17.5	a = 0.006360
at oP = 18.0	a = 0.005709	at oP = 18.5	a = 0.004908
at oP = 19.0	a = 0.003753	at oP = 19.5	a = 0.002325
at oP = 20.0	a = 0.001458	at oP = 20.5	a = -0.000052

average a is 0.006881

enter starting oP: minimum oP where a becomes reasonably constant

5

average w is 0.057436

re-estimate of a is 0.0069 average Rp is 5217.

estimated Icorr is 0.00166878

re-estimate of w is 0.05757037

initial Ecorr,ba,bc,Icorr and Rp are

-255.11 35.73 45.43 0.001665 5217.

initial relative RMS error is 0.082915

enter 1 if acceptable

1

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

5.559737E-04	-2.110495E-05	2.477400E-05	-6.371798E-06
5.498207E-04	-2.090320E-05	2.471899E-05	-6.459422E-06
5.500678E-04	-2.091288E-05	2.472174E-05	-6.454742E-06
5.500761E-04	-2.092529E-05	2.471082E-05	-6.436885E-06

gradient: dEcorr,dba,dbc,dIcorr

5.500043E-04	-2.091514E-05	2.471674E-05	-6.449062E-06
--------------	---------------	--------------	---------------

Jacobian before symmetry check

5.969370E-01	-1.836815E-02	6.341725E-03	7.173367E-03
-1.898122E-02	1.193507E-03	2.252136E-04	-1.015496E-03
5.726088E-03	2.246452E-04	4.996309E-04	-5.922402E-04
7.726374E-03	-1.036022E-03	-5.680192E-04	1.217643E-03

Jacobian matrix

5.969370E-01	-1.867469E-02	6.033906E-03	7.449870E-03
-1.867469E-02	1.193507E-03	2.249294E-04	-1.025759E-03
6.033906E-03	2.249294E-04	4.996309E-04	-5.801297E-04
7.449870E-03	-1.025759E-03	-5.801297E-04	1.217643E-03

initial correction vector

5.176613E-03	3.579467E-01	1.037736E-01	3.246053E-01
--------------	--------------	--------------	--------------

error overshoot: reduced correction by half

Ecorr ba bc Icorr Rp
-255.77 42.12 47.78 0.001935 5024.

relative RMS error now = 0.073427

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

5.857900E-04	-1.760131E-05	1.349978E-05	3.512548E-06
5.800501E-04	-1.744107E-05	1.344255E-05	3.444029E-06
5.802585E-04	-1.744835E-05	1.344500E-05	3.447605E-06
5.802477E-04	-1.746020E-05	1.343397E-05	3.466087E-06

gradient: dEcorr,dba,dbc,dIcorr

5.801946E-04	-1.745080E-05	1.344010E-05	3.453597E-06
--------------	---------------	--------------	--------------

Jacobian before symmetry check

5.595313E-01	-1.445951E-02	6.390474E-03	5.306356E-03
-1.505189E-02	9.726819E-04	2.443812E-04	-9.401674E-04
5.967911E-03	2.443358E-04	4.901608E-04	-6.131472E-04
5.895069E-03	-9.568566E-04	-5.992291E-04	1.248921E-03

Jacobian matrix

5.595313E-01	-1.475570E-02	6.179192E-03	5.600712E-03
-1.475570E-02	9.726819E-04	2.443585E-04	-9.485120E-04
6.179192E-03	2.443585E-04	4.901608E-04	-6.061882E-04
5.600712E-03	-9.485120E-04	-6.061882E-04	1.248921E-03

initial correction vector

3.298722E-04	-1.137024E-01	-2.174872E-01	-1.961592E-01
--------------	---------------	---------------	---------------

Ecorr ba bc Icorr Rp

-255.86 37.33 37.39 0.001556 5215.

relative RMS error now = 0.063821

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

-4.108566E-04	2.253072E-06	-1.302219E-05	9.335019E-06
-4.172401E-04	2.412895E-06	-1.313840E-05	9.305170E-06
-4.169753E-04	2.404754E-06	-1.313292E-05	9.306781E-06
-4.170642E-04	2.393229E-06	-1.314816E-05	9.327106E-06

gradient: dEcorr,dba,dbc,dIcorr

-4.170889E-04	2.402509E-06	-1.314064E-05	9.314424E-06
---------------	--------------	---------------	--------------

Jacobian before symmetry check

6.232323E-01	-1.511435E-02	1.136286E-02	2.469824E-03
-1.494373E-02	1.038634E-03	2.245457E-04	-9.280228E-04

1.184525E-02 2.244406E-04 7.718995E-04 -7.513677E-04
2.059539E-03 -9.254222E-04 -7.642825E-04 1.268177E-03

Jacobian matrix

6.232323E-01 -1.502904E-02 1.160406E-02 2.264682E-03
-1.502904E-02 1.038634E-03 2.244931E-04 -9.267225E-04
1.160406E-02 2.244931E-04 7.718995E-04 -7.578251E-04
2.264682E-03 -9.267225E-04 -7.578251E-04 1.268177E-03

initial correction vector

1.356052E-03 -6.445559E-02 -9.785674E-02 -1.153437E-01

Ecorr ba bc Icorr Rp

-256.20 34.92 33.73 0.001376 5415.

relative RMS error now = 0.060482

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

-5.223650E-05 1.043271E-06 -2.257839E-06 1.009662E-06
-5.892037E-05 1.223551E-06 -2.384136E-06 9.728229E-07
-5.862422E-05 1.213895E-06 -2.377776E-06 9.749527E-07
-5.872406E-05 1.202092E-06 -2.393776E-06 9.949578E-07

gradient: dEcorr,dba,dbc,dIcorr

-5.875171E-05 1.211821E-06 -2.386212E-06 9.825310E-07

Jacobian before symmetry check

6.515212E-01 -1.686581E-02 1.274907E-02 2.765046E-03
-1.685510E-02 1.172992E-03 2.073932E-04 -9.729263E-04
1.283736E-02 2.076632E-04 8.436587E-04 -7.563330E-04
2.713062E-03 -9.708174E-04 -7.578336E-04 1.242671E-03

Jacobian matrix

6.515212E-01 -1.686045E-02 1.279321E-02 2.739054E-03
-1.686045E-02 1.172992E-03 2.075282E-04 -9.718718E-04
1.279321E-02 2.075282E-04 8.436587E-04 -7.570833E-04
2.739054E-03 -9.718718E-04 -7.570833E-04 1.242671E-03

initial correction vector

-8.115192E-05 -7.518587E-03 1.824736E-04 -6.380777E-03

Ecorr ba bc Icorr Rp

-256.18 34.66 33.74 0.001367 5430.

relative RMS error now = 0.060409

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

7.319809E-06	-2.275095E-07	1.263722E-07	6.491456E-08
6.366540E-07	-4.393530E-08	2.458801E-09	2.510726E-08
9.344562E-07	-5.378701E-08	8.758978E-09	2.740973E-08
8.383090E-07	-6.564670E-08	-7.108129E-09	4.729421E-08

gradient: dEcorr,dba,dbc,dIcorr

8.084107E-07	-5.585315E-08	3.919034E-10	3.490874E-08
--------------	---------------	--------------	--------------

Jacobian before symmetry check

6.511399E-01	-1.717567E-02	1.260455E-02	2.989827E-03
-1.716563E-02	1.191785E-03	2.066145E-04	-9.793553E-04
1.259803E-02	2.066898E-04	8.367074E-04	-7.500033E-04
3.000582E-03	-9.801482E-04	-7.499011E-04	1.238547E-03

Jacobian matrix

6.511399E-01	-1.717065E-02	1.260129E-02	2.995205E-03
-1.717065E-02	1.191785E-03	2.066521E-04	-9.797518E-04
1.260129E-02	2.066521E-04	8.367074E-04	-7.499522E-04
2.995205E-03	-9.797518E-04	-7.499522E-04	1.238547E-03

initial correction vector

1.476836E-06	2.615836E-04	1.524016E-04	2.674494E-04
--------------	--------------	--------------	--------------

Ecorr	ba	bc	Icorr	Rp
-256.18	34.67	33.74	0.001368	5430.

relative RMS error now = 0.060409

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

6.511249E-06	-1.709626E-07	1.255378E-07	3.002279E-08
-1.702792E-07	1.248890E-08	1.639860E-09	-9.728382E-09
1.275062E-07	2.649870E-09	7.948902E-09	-7.441514E-09
3.123887E-08	-9.218411E-09	-7.931849E-09	1.246110E-08

gradient: dEcorr,dba,dbc,dIcorr

1.336048E-09	5.812808E-10	-4.255068E-10	6.548362E-11
--------------	--------------	---------------	--------------

Jacobian before symmetry check

6.509913E-01	-1.716153E-02	1.261701E-02	2.990282E-03
-1.715439E-02	1.190762E-03	2.068589E-04	-9.799692E-04
1.259633E-02	2.065367E-04	8.374409E-04	-7.506342E-04
2.995731E-03	-9.793865E-04	-7.506998E-04	1.239562E-03

Jacobian matrix

6.509913E-01	-1.715796E-02	1.260667E-02	2.993006E-03
-1.715796E-02	1.190762E-03	2.066978E-04	-9.796779E-04
1.260667E-02	2.066978E-04	8.374409E-04	-7.506670E-04
2.993006E-03	-9.796779E-04	-7.506670E-04	1.239562E-03

initial correction vector

-1.812548E-07 -1.093269E-05 -3.205017E-06 -1.019667E-05

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

Ecorr ba bc Icorr Rp

-256.18 34.67 33.74 0.001368 5430.

relative RMS error now = 0.060409

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

6.512033E-06 -1.709817E-07 1.255612E-07 3.001824E-08

-1.694675E-07 1.246798E-08 1.663622E-09 -9.732048E-09

1.283875E-07 2.625029E-09 7.973966E-09 -7.442935E-09

3.205787E-08 -9.240864E-09 -7.908795E-09 1.245829E-08

gradient: dEcorr,dba,dbc,dIcorr

2.169145E-09 5.580887E-10 -4.025669E-10 6.369305E-11

Jacobian before symmetry check

6.509863E-01 -1.716367E-02 1.262183E-02 2.988872E-03

-1.715398E-02 1.190989E-03 2.066940E-04 -9.798953E-04

1.259638E-02 2.066189E-04 8.376533E-04 -7.506229E-04

2.995455E-03 -9.795741E-04 -7.506628E-04 1.239459E-03

Jacobian matrix

6.509863E-01 -1.715882E-02 1.260910E-02 2.992164E-03

-1.715882E-02 1.190989E-03 2.066565E-04 -9.797347E-04

1.260910E-02 2.066565E-04 8.376533E-04 -7.506429E-04

2.992164E-03 -9.797347E-04 -7.506429E-04 1.239459E-03

initial correction vector

-1.811797E-07 -1.083091E-05 -3.162521E-06 -1.009062E-05

error overshoot: reduced correction by half

error overshoot: reduced correction by half

Ecorr ba bc Icorr Rp

-256.18 34.67 33.74 0.001368 5430.

relative RMS error now = 0.060409

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

6.511890E-06	-1.711419E-07	1.256683E-07	3.000849E-08
-1.695817E-07	1.230893E-08	1.772985E-09	-9.744071E-09
1.282406E-07	2.465868E-09	8.082171E-09	-7.454048E-09
3.196010E-08	-9.402925E-09	-7.800276E-09	1.244962E-08

gradient: dEcorr,dba,dbc,dIcorr

2.053639E-09	3.975629E-10	-2.939675E-10	5.320544E-11
--------------	--------------	---------------	--------------

Jacobian before symmetry check

6.509837E-01	1.716353E-02	1.261869E-02	2.990646E-03
-1.715395E-02	1.191137E-03	2.068305E-04	-9.800488E-04
1.259622E-02	2.066953E-04	8.376139E-04	-7.506309E-04
2.995529E-03	-9.797276E-04	-7.507254E-04	1.239641E-03

Jacobian matrix

6.509837E-01	-1.715874E-02	1.260746E-02	2.993087E-03
-1.715874E-02	1.191137E-03	2.067629E-04	-9.798882E-04
1.260746E-02	2.067629E-04	8.376139E-04	-7.506781E-04
2.993087E-03	-9.798882E-04	-7.506781E-04	1.239641E-03

initial correction vector

-1.353360E-07	-7.883587E-06	-2.179073E-06	-7.267384E-06
---------------	---------------	---------------	---------------

Ecorr ba bc Icorr Rp

-256.18	34.67	33.74	0.001368	5430.
---------	-------	-------	----------	-------

relative RMS error now = 0.060409

enter 1 to continue : 99 to quit

1

gradient at each parameter + del

6.510722E-06	-1.715841E-07	1.259860E-07	2.996003E-08
-1.708609E-07	1.187419E-08	2.090268E-09	-9.797049E-09
1.269004E-07	2.031868E-09	8.396494E-09	-7.505719E-09
3.064815E-08	-9.835617E-09	-7.482445E-09	1.239516E-08

gradient: dEcorr,dba,dbc,dIcorr

7.985364E-10	-3.859668E-11	2.371081E-11	1.193712E-12
--------------	---------------	--------------	--------------

Jacobian before symmetry check

6.509923E-01	-1.716594E-02	1.261019E-02	2.984962E-03
-1.715455E-02	1.191279E-03	2.070465E-04	-9.797020E-04
1.259623E-02	2.066557E-04	8.372783E-04	-7.506156E-04
2.995884E-03	-9.798242E-04	-7.506913E-04	1.239397E-03

Jacobian matrix

6.509923E-01	-1.716025E-02	1.260321E-02	2.990423E-03
-1.716025E-02	1.191279E-03	2.068511E-04	-9.797631E-04
1.260321E-02	2.068511E-04	8.372783E-04	-7.506534E-04

2.990423E-03 -9.797631E-04 -7.506534E-04 1.239397E-03

initial correction vector

2.438097E-09 5.157828E-07 3.653581E-07 6.221716E-07

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

error overshoot: reduced correction by half

no further improvement available

Ecorr	ba	bc	Icorr	Rp
-256.18	34.67	33.74	0.001368	5430.

relative RMS error now = 0.060409

enter 1 to continue : 99 to quit

99

relative RMS error now = 0.060409

Ecorr = -256.18 millivolts

ba = 34.67 +/- 4.640 millivolts

bc = 33.74 +/- 2.949 millivolts

Rp = 5430. +/- 460.8 ohms

Icorr = 0.001368 +/- 0.00011417 millamps/cm²

TAPE number ?

5

FILE number ?

4

APPENDIX G

PLOTS

This is the format of a PLOTS file as it is output when running CORROS. This file will be accepted when running <Staal>SAPLTF² and will result in graphic presentation of the data.

@type plots..

```
CSET 1 0
CSET 2 0
LSET 1 6
LSET 2 0
PSET 2.95.
DENSIT 0
LABEL 1
potential (millivolts vs. SCE)
LABEL 2
current density (microamps/cm!SUP;2!BAK;)
LABEL -4217
RMS error = 0.0604
LABEL -4211
Ecorr = -256.18 mV
LABEL -4205
Icorr = 1.36766 !FNT13;m!FNT3;A/cm!SUP;2!BAK;
LEGEND
data
fitted curve
CURVE
-2.760000E+02 -5.727000E+00
-2.755000E+02 -5.063000E+00
-2.750000E+02 -4.662000E+00
-2.745000E+02 -4.388000E+00
-2.740000E+02 -4.046000E+00
-2.735000E+02 -3.811000E+00
-2.730000E+02 -3.616000E+00
-2.725000E+02 -3.440000E+00
-2.720000E+02 -3.284000E+00
-2.715000E+02 -3.147000E+00
-2.710000E+02 -3.020000E+00
-2.705000E+02 -2.883000E+00
-2.700000E+02 -2.766000E+00
-2.695000E+02 -2.658000E+00
-2.690000E+02 -2.551000E+00
-2.685000E+02 -2.453000E+00
-2.680000E+02 -2.355000E+00
-2.675000E+02 -2.267000E+00
```

-2.670000E+02	-2.169000E+00
-2.665000E+02	-2.081000E+00
-2.660000E+02	-1.993000E+00
-2.655000E+02	-1.905000E+00
-2.650000E+02	-1.827000E+00
-2.645000E+02	-1.739000E+00
-2.640000E+02	-1.651000E+00
-2.635000E+02	-1.573000E+00
-2.630000E+02	-1.485000E+00
-2.625000E+02	-1.397000E+00
-2.620000E+02	-1.299000E+00
-2.615000E+02	-1.221000E+00
-2.610000E+02	-1.133000E+00
-2.605000E+02	-1.036000E+00
-2.600000E+02	-9.490000E-01
-2.595000E+02	-8.601000E-01
-2.590000E+02	-7.633000E-01
-2.585000E+02	-6.666000E-01
-2.580000E+02	-5.669000E-01
-2.575000E+02	-4.662000E-01
-2.570000E+02	-3.675000E-01
-2.565000E+02	-2.619000E-01
-2.560000E+02	-1.612000E-01
-2.555000E+02	-6.998000E-02
-2.550000E+02	2.023000E-02
-2.545000E+02	1.094000E-01
-2.540000E+02	2.003000E-01
-2.535000E+02	2.922000E-01
-2.530000E+02	3.743000E-01
-2.525000E+02	4.623000E-01
-2.520000E+02	5.629000E-01
-2.515000E+02	6.685000E-01
-2.510000E+02	7.780000E-01
-2.505000E+02	8.806000E-01
-2.500000E+02	9.920000E-01
-2.495000E+02	1.104000E+00
-2.490000E+02	1.212000E+00
-2.485000E+02	1.319000E+00
-2.480000E+02	1.436000E+00
-2.475000E+02	1.554000E+00
-2.470000E+02	1.671000E+00
-2.465000E+02	1.818000E+00
-2.460000E+02	1.974000E+00
-2.455000E+02	2.130000E+00
-2.450000E+02	2.296000E+00
-2.445000E+02	2.472000E+00
-2.440000E+02	2.629000E+00
-2.435000E+02	2.795000E+00
-2.430000E+02	2.932000E+00

-2.425000E+02	3.069000E+00
-2.420000E+02	3.205000E+00
-2.415000E+02	3.323000E+00
-2.410000E+02	3.440000E+00
-2.405000E+02	3.567000E+00
-2.400000E+02	3.694000E+00
-2.395000E+02	3.821000E+00
-2.390000E+02	3.948000E+00
-2.385000E+02	4.085000E+00
-2.380000E+02	4.232000E+00
-2.375000E+02	4.378000E+00
-2.370000E+02	4.525000E+00
-2.365000E+02	4.652000E+00
-2.360000E+02	4.779000E+00
-2.355000E+02	4.906000E+00
-2.350000E+02	5.053000E+00
-2.345000E+02	5.209000E+00

CURVE

-2.760000E+02	-4.920840E+00
-2.755000E+02	-4.731086E+00
-2.750000E+02	-4.546863E+00
-2.745000E+02	-4.367956E+00
-2.740000E+02	-4.194163E+00
-2.735000E+02	-4.025277E+00
-2.730000E+02	-3.861105E+00
-2.725000E+02	-3.701454E+00
-2.720000E+02	-3.546143E+00
-2.715000E+02	-3.394990E+00
-2.710000E+02	-3.247819E+00
-2.705000E+02	-3.104461E+00
-2.700000E+02	-2.964752E+00
-2.695000E+02	-2.828527E+00
-2.690000E+02	-2.695630E+00
-2.685000E+02	-2.565907E+00
-2.680000E+02	-2.439209E+00
-2.675000E+02	-2.315389E+00
-2.670000E+02	-2.194305E+00
-2.665000E+02	-2.075816E+00
-2.660000E+02	-1.959787E+00
-2.655000E+02	-1.846082E+00
-2.650000E+02	-1.734574E+00
-2.645000E+02	-1.625131E+00
-2.640000E+02	-1.517630E+00
-2.635000E+02	-1.411944E+00
-2.630000E+02	-1.307954E+00
-2.625000E+02	-1.205541E+00
-2.620000E+02	-1.104587E+00
-2.615000E+02	-1.004976E+00
-2.610000E+02	-9.065937E-01

-2.605000E+02	-8.093285E-01
-2.600000E+02	-7.130686E-01
-2.595000E+02	-6.177044E-01
-2.590000E+02	-5.231267E-01
-2.585000E+02	-4.292280E-01
-2.580000E+02	-3.359011E-01
-2.575000E+02	-2.430398E-01
-2.570000E+02	-1.505386E-01
-2.565000E+02	-5.829243E-02
-2.560000E+02	3.380389E-02
-2.555000E+02	1.258544E-01
-2.550000E+02	2.179627E-01
-2.545000E+02	3.102341E-01
-2.540000E+02	4.027723E-01
-2.535000E+02	4.956825E-01
-2.530000E+02	5.890696E-01
-2.525000E+02	6.830383E-01
-2.520000E+02	7.776944E-01
-2.515000E+02	8.731452E-01
-2.510000E+02	9.694982E-01
-2.505000E+02	1.066861E+00
-2.500000E+02	1.165343E+00
-2.495000E+02	1.265055E+00
-2.490000E+02	1.366108E+00
-2.485000E+02	1.468617E+00
-2.480000E+02	1.572696E+00
-2.475000E+02	1.678460E+00
-2.470000E+02	1.786029E+00
-2.465000E+02	1.895523E+00
-2.460000E+02	2.007064E+00
-2.455000E+02	2.120775E+00
-2.450000E+02	2.236786E+00
-2.445000E+02	2.355223E+00
-2.440000E+02	2.476220E+00
-2.435000E+02	2.599910E+00
-2.430000E+02	2.726434E+00
-2.425000E+02	2.855928E+00
-2.420000E+02	2.988540E+00
-2.415000E+02	3.124414E+00
-2.410000E+02	3.263705E+00
-2.405000E+02	3.406563E+00
-2.400000E+02	3.553150E+00
-2.395000E+02	3.703626E+00
-2.390000E+02	3.858161E+00
-2.385000E+02	4.016924E+00
-2.380000E+02	4.180092E+00
-2.375000E+02	4.347845E+00
-2.370000E+02	4.520369E+00
-2.365000E+02	4.697855E+00

-2.360000E+02	4.880500E+00
-2.355000E+02	5.068506E+00
-2.350000E+02	5.262080E+00
-2.345000E+02	5.461438E+00

APPENDIX H

LIST

This is the format of a LIST file as it is output when running CORROS. This file is appended each time CORROS is run and may therefore serve as a summary of results for a group of polarization resistance measurements.

@type list..

File 4
relative RMS error = 0.060409
Ecorr = -256.18 millivolts
Icorr = 1.36766 +/- 0.11417 microamps/cm²
ba = 34.67 +/- 4.64 millivolts
bc = 33.74 +/- 2.95 millivolts
Rp = 5.43 +/- 0.46 kohms

APPENDIX I

Donahue's³⁵ Data

@type tp1tst.dat

this is a file containing data from a paper
on corrosion analysis by Donahue³⁵

RESULTS

ECORR = 200.0 mV

0.	-47.E6
0.02	-32.E6
0.04	-21.E6
0.06	-15.E6
0.08	-10.E6
0.1	-6.8E6
0.11	-5.6E6
0.12	-4.6E6
0.13	-3.9E6
0.14	-3.1E6
0.15	-2.6E6
0.155	-2.3E6
0.16	-2.1E6
0.165	-1.8E6
0.17	-1.6E6
0.175	-1.37E6
0.18	-1.15E6
0.185	-0.90E6
0.19	-0.64E6
0.195	-0.35E6
0.2	0.00
0.205	0.42E6
0.21	0.96E6
0.215	1.76E6
0.22	2.50E6
0.225	3.8E6
0.23	5.1E6
0.235	7.2E6
0.24	9.5E6
0.245	13.E6
0.25	18.E6
0.26	32.E6
0.27	57.E6
0.28	100.E6

APPENDIX I

Simplified CORROS Analysis of Donahue's³⁵ Data

@ex@corrode

PASCAL: CORROS
FORTRAN: CORSUB
SPLINE
SEVAL
DECOMP
SOLVE
LINK: Loading
[LNKXCT CORROS execution]
INPUT :
OUTPUT :
PLOTS :
DATA : tp1tst.dat
LIST :

Would you like to read the introduction ? yes = 1
2

do you want a detailed run ? yes = 1
2

this is a test file containing data from a paper on corrosion analysis by Donahue³⁵
RESULTS
ECORR = 200.0 mV *** end of tape header ***

34 data points read in

do you want to see the data tabulated? yes = 1
2

estimated Ecorr is 200.00
range around Ecorr reduced to 80.0
27 points in range

at oP = 5.0	a = 0.018232	at oP = 5.5	a = 0.017987
at oP = 6.0	a = 0.017821	at oP = 6.5	a = 0.017768
at oP = 7.0	a = 0.017836	at oP = 7.5	a = 0.018021
at oP = 8.0	a = 0.018312	at oP = 8.5	a = 0.018697
at oP = 9.0	a = 0.019161	at oP = 9.5	a = 0.019691
at oP = 10.0	a = 0.020273	at oP = 10.5	a = 0.020884
at oP = 11.0	a = 0.021464	at oP = 11.5	a = 0.021969
at oP = 12.0	a = 0.022374	at oP = 12.5	a = 0.022664
at oP = 13.0	a = 0.022835	at oP = 13.5	a = 0.022886
at oP = 14.0	a = 0.022821	at oP = 14.5	a = 0.022642

at oP = 15.0	a = 0.022356	at oP = 15.5	a = 0.021975
at oP = 16.0	a = 0.021540	at oP = 16.5	a = 0.021093
at oP = 17.0	a = 0.020662	at oP = 17.5	a = 0.020272
at oP = 18.0	a = 0.019941	at oP = 18.5	a = 0.019681
at oP = 19.0	a = 0.019503	at oP = 19.5	a = 0.019413
at oP = 20.0	a = 0.019413	at oP = 20.5	a = 0.019499
at oP = 21.0	a = 0.019641	at oP = 21.5	a = 0.019811
at oP = 22.0	a = 0.019986	at oP = 22.5	a = 0.020149
at oP = 23.0	a = 0.020286	at oP = 23.5	a = 0.020387
at oP = 24.0	a = 0.020444	at oP = 24.5	a = 0.020451
at oP = 25.0	a = 0.020404	at oP = 25.5	a = 0.020304
at oP = 26.0	a = 0.020168	at oP = 26.5	a = 0.020012
at oP = 27.0	a = 0.019850	at oP = 27.5	a = 0.019696
at oP = 28.0	a = 0.019558	at oP = 28.5	a = 0.019446
at oP = 29.0	a = 0.019365	at oP = 29.5	a = 0.019322
at oP = 30.0	a = 0.019321	at oP = 30.5	a = 0.019361
at oP = 31.0	a = 0.019430	at oP = 31.5	a = 0.019516
at oP = 32.0	a = 0.019606	at oP = 32.5	a = 0.019692
at oP = 33.0	a = 0.019764	at oP = 33.5	a = 0.019817
at oP = 34.0	a = 0.019844	at oP = 34.5	a = 0.019841
at oP = 35.0	a = 0.019804	at oP = 35.5	a = 0.019732
at oP = 36.0	a = 0.019634	at oP = 36.5	a = 0.019520
at oP = 37.0	a = 0.019398	at oP = 37.5	a = 0.019276
at oP = 38.0	a = 0.019160	at oP = 38.5	a = 0.019057
at oP = 39.0	a = 0.018971	at oP = 39.5	a = 0.018907
at oP = 40.0	a = 0.018867	at oP = 40.5	a = 0.018854
at oP = 41.0	a = 0.018864	at oP = 41.5	a = 0.018892
at oP = 42.0	a = 0.018933	at oP = 42.5	a = 0.018983
at oP = 43.0	a = 0.019039	at oP = 43.5	a = 0.019096
at oP = 44.0	a = 0.019151	at oP = 44.5	a = 0.019202
at oP = 45.0	a = 0.019245	at oP = 45.5	a = 0.019279
at oP = 46.0	a = 0.019304	at oP = 46.5	a = 0.019322
at oP = 47.0	a = 0.019334	at oP = 47.5	a = 0.019341
at oP = 48.0	a = 0.019345	at oP = 48.5	a = 0.019347
at oP = 49.0	a = 0.019347	at oP = 49.5	a = 0.019347
at oP = 50.0	a = 0.019349	at oP = 50.5	a = 0.019352
at oP = 51.0	a = 0.019356	at oP = 51.5	a = 0.019363
at oP = 52.0	a = 0.019371	at oP = 52.5	a = 0.019379
at oP = 53.0	a = 0.019389	at oP = 53.5	a = 0.019399
at oP = 54.0	a = 0.019410	at oP = 54.5	a = 0.019420
at oP = 55.0	a = 0.019430	at oP = 55.5	a = 0.019440
at oP = 56.0	a = 0.019449	at oP = 56.5	a = 0.019456
at oP = 57.0	a = 0.019462	at oP = 57.5	a = 0.019467
at oP = 58.0	a = 0.019469	at oP = 58.5	a = 0.019469
at oP = 59.0	a = 0.019467	at oP = 59.5	a = 0.019461
at oP = 60.0	a = 0.019453	at oP = 60.5	a = 0.019441
at oP = 61.0	a = 0.019427	at oP = 61.5	a = 0.019411
at oP = 62.0	a = 0.019393	at oP = 62.5	a = 0.019373
at oP = 63.0	a = 0.019353	at oP = 63.5	a = 0.019333

at oP = 64.0 a = 0.019312 at oP = 64.5 a = 0.019292
 at oP = 65.0 a = 0.019273 at oP = 65.5 a = 0.019254
 at oP = 66.0 a = 0.019236 at oP = 66.5 a = 0.019220
 at oP = 67.0 a = 0.019205 at oP = 67.5 a = 0.019192
 at oP = 68.0 a = 0.019181 at oP = 68.5 a = 0.019172
 at oP = 69.0 a = 0.019165 at oP = 69.5 a = 0.019160
 at oP = 70.0 a = 0.019158 at oP = 70.5 a = 0.019158
 at oP = 71.0 a = 0.019160 at oP = 71.5 a = 0.019164
 at oP = 72.0 a = 0.019170 at oP = 72.5 a = 0.019176
 at oP = 73.0 a = 0.019184 at oP = 73.5 a = 0.019192
 at oP = 74.0 a = 0.019200 at oP = 74.5 a = 0.019208
 at oP = 75.0 a = 0.019216 at oP = 75.5 a = 0.019224
 at oP = 76.0 a = 0.019231 at oP = 76.5 a = 0.019237
 at oP = 77.0 a = 0.019243 at oP = 77.5 a = 0.019247
 at oP = 78.0 a = 0.019250 at oP = 78.5 a = 0.019251
 at oP = 79.0 a = 0.019251 at oP = 79.5 a = 0.019248
 at oP = 80.0 a = 0.019244 average a is 0.019598

enter starting oP: minimum oP where a becomes reasonably constant

5

data too scattered at 5.0 to give a w

average w is 0.039641

re-estimate of a is 0.0196 average Rp is 13.

estimated Icorr is 0.95187600

re-estimate of w is 0.03980096

initial Ecorr,ba,bc,Icorr and Rp are

200.00 38.76 113.99 0.948041 13.

initial relative RMS error is 0.083927

enter 1 if acceptable

1

enter 1 to continue : 99 to quit

1

Ecorr ba bc Icorr Rp
 202.49 37.07 106.57 0.837627 14.

relative RMS error now = 0.046343

enter 1 to continue : 99 to quit

1

Ecorr ba bc Icorr Rp
 198.95 40.82 122.23 1.038288 13.

relative RMS error now = 0.012931

enter 1 to continue : 99 to quit

1

Ecorr ba bc Icorr Rp
 199.38 40.44 121.38 1.025582 13.

relative RMS error now = 0.008776

enter 1 to continue : 99 to quit

1

Ecorr ba bc Icorr Rp
199.47 40.53 122.10 1.034750 13.
relative RMS error now = 0.005835
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
200.00 40.57 126.53 1.070331 12.
relative RMS error now = 0.005716
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
199.97 40.57 126.43 1.068972 12.
relative RMS error now = 0.005692
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
200.01 40.57 126.83 1.072053 12.
relative RMS error now = 0.005689
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
200.01 40.57 126.75 1.071465 12.
relative RMS error now = 0.005689
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
200.00 40.57 126.72 1.071242 12.
relative RMS error now = 0.005689
enter 1 to continue : 99 to quit
1

Ecorr ba bc Icorr Rp
200.00 40.57 126.70 1.071044 12.
relative RMS error now = 0.005689
enter 1 to continue : 99 to quit
1

no further improvement available

Ecorr ba bc Icorr Rp
200.00 40.57 126.70 1.071044 12.
relative RMS error now = 0.005689
enter 1 to continue : 99 to quit

99

relative RMS error now = 0.005689

Ecorr = 200.00 millivolts

ba = 40.57 +/- 0.712 millivolts

bc = 126.70 +/- 13.147 millivolts

Rp = 12. +/- 0.4 ohms

Icorr = 1.071044 +/- 0.03308610 millamps/cm²

TAPE number ?

1

FILE number ?

1

APPENDIX K

PLOTS From CORROS Analysis of Donahue's³⁵ Data

@type plots..

CSET 1 0

CSET 2 0

LSET 1 6

LSET 2 0

PSET 2. 95.

DENSIT 0

LABEL 1

potential (millivolts vs. SCE)

LABEL 2

current density (microamps/cm!SUP;2!BAK;)

LABEL -4217

RMS error = 0.0057

LABEL -4211

Ecorr = 200.00 mV

LABEL -4205

Icorr = 1071.04400 !FNT13;m!FNT3;A/cm!SUP;2!BAK;

LEGEND

data

fitted curve

CURVE

1.200000E+02	-4.600000E+03
1.300000E+02	-3.900000E+03
1.400000E+02	-3.100000E+03
1.500000E+02	-2.600000E+03
1.550000E+02	-2.300000E+03
1.600000E+02	-2.100000E+03
1.650000E+02	-1.800000E+03
1.700000E+02	-1.600000E+03
1.750000E+02	-1.370000E+03
1.800000E+02	-1.150000E+03
1.850000E+02	-9.000000E+02
1.900000E+02	-6.400000E+02
1.950000E+02	-3.500000E+02
2.000000E+02	0.000000E+00
2.050000E+02	4.200000E+02
2.100000E+02	9.600000E+02
2.150000E+02	1.760000E+03
2.200000E+02	2.500000E+03
2.250000E+02	3.800000E+03
2.300000E+02	5.100000E+03
2.350000E+02	7.200000E+03

2.400000E+02	9.500000E+03
2.450000E+02	1.300000E+04
2.500000E+02	1.800000E+04
2.600000E+02	3.200000E+04
2.700000E+02	5.700000E+04
2.800000E+02	1.000000E+05

CURVE

1.200000E+02	-4.572435E+03
1.300000E+02	-3.801960E+03
1.400000E+02	-3.151405E+03
1.500000E+02	-2.594638E+03
1.550000E+02	-2.343237E+03
1.600000E+02	-2.105133E+03
1.650000E+02	-1.876370E+03
1.700000E+02	-1.652412E+03
1.750000E+02	-1.427900E+03
1.800000E+02	-1.196315E+03
1.850000E+02	-9.495513E+02
1.900000E+02	-6.773440E+02
1.950000E+02	-3.665198E+02
2.000000E+02	-9.176887E-04
2.050000E+02	4.445119E+02
2.100000E+02	9.962865E+02
2.150000E+02	1.693876E+03
2.200000E+02	2.588196E+03
2.250000E+02	3.746600E+03
2.300000E+02	5.258308E+03
2.350000E+02	7.241592E+03
2.400000E+02	9.853335E+03
2.450000E+02	1.330172E+04
2.500000E+02	1.786308E+04
2.600000E+02	3.191243E+04
2.700000E+02	5.662904E+04
2.800000E+02	1.001740E+05

APPENDIX L

LIST From CORROS Analysis of Donahue's³⁵ Data

@type list.

File 1
relative RMS error = 0.005689
Ecorr = 200.00 millivolts
Icorr = 1071.04400 +/- 33.08610 microamps/cm²
ba = 40.57 +/- 0.71 millivolts
bc = 126.70 +/- 13.15 millivolts
Rp = 0.01 +/- 0.00 kohms

REFERENCES

1. Staal, P.R., Defence Research Establishment Atlantic, informal communication.
2. Staal, P.R., Defence Research Establishment Atlantic, informal communication.
3. Ailor, W.H., ed., Handbook on Corrosion Testing and Evaluation, John Wiley and Sons, Inc., New York, p. 119, 1971.
4. Wagner, C.W. and Traud, W., "On the Significance of Corrosion Processes Through Superposition of Electrochemical Partial Processes and on the Formation of Potential on a Mixed Electrode", Z. Elektrochem., 44, p. 391, 1938.
5. LaQue, F.L., Baboian, R., Kruger, J., Uhlig, H.H., Verink, E.D., Brown, B.F., Hackerman, N. and Streicher, M., "Debate on the Applicability of the Use of Electrochemical Techniques for Corrosion Studies", Educational Lecture, CORROSION/81, The International Corrosion Forum Sponsored by the National Association of Corrosion Engineers, Toronto Ontario Canada, April 6-10, 1981.
6. LaQue, F.L., "Guest Editorial: Shortcuts to Prediction of Corrosion Rates - Do They Yield Valid Results?", Corrosion, Vol. 35, No. 5, p. i, 1979.
7. LaQue, F.L., Discussion, "A Potentiometric Study of the Corrosion of Ni-Resist in Seawater", Corrosion, 38, p. 234, 1982.
8. Uhlig, H.H., "Guest Editorial: Knowing, Competence, and Optimism Applied to Corrosion Problems", Corrosion, Vol. 35, No. 9, p. i, 1979.
9. Fontana, M.G., "Corrosion Prevention, Past, Present, and Future", Materials Performance, 15, p. 49, 1976.
10. 1983 Annual Book of ASTM Standards, Section 3: Metals Test Methods and Analytical Procedures, Volume 03.02: Metal Corrosion, Erosion, and Wear, G5-82, American Society for Testing and Materials, Philadelphia, PA., p. 122, 1983.
11. Stern, M. and Geary, A.L., "Electrochemical Polarization I. A Theoretical Analysis of the Shape of Polarization Curves", J. Electrochem. Soc., 104, p. 56, 1957.
12. Stern, M., "A Method for Determining Corrosion Rates from Linear Polarization Data", Corrosion, 14, p. 440t, 1958.
13. Oldham, K. and Mansfeld, F., Technical Note, "On the So-Called Linear Polarization Method for Measurement of Corrosion Rates", Corrosion, 27, p. 434, 1971.
14. Mansfeld, F. and Oldham, K., Comment, "Some Limitations of the Linear Polarization Techniques in Evaluating Corrosion Behavior", Corrosion, 26, p. 207, 1970.
15. Mansfeld, F., Discussion, "On the Shape of Some Polarization Curves in the Vicinity of the Corrosion Potential", Corrosion, 30, p. 320, 1974.

16. Jones, D.A., Discussion, "On the So-Called Polarization Method for Measurement of Corrosion Rates", *Corrosion*, 28, p. 180, 1972.
17. Oldham, K.B. and Mansfeld, F., "Corrosion Rates From Polarization Curves: A New Method", *Corrosion Science*, 13, p. 813, 1973.
18. Mansfeld, F., "Simultaneous Determination of Instantaneous Corrosion Rates and Tafel Slopes from Polarization Resistance Measurements", *J. Electrochem. Soc.*, 120, p. 515, 1973.
19. Mansfeld, F., "Tafel Slopes and Corrosion Rates from Polarization Resistance Measurements", *Corrosion*, 29, p. 397, 1973.
20. Gerchakov, S.M., Udey, L.R. and Mansfeld, F., "An Improved Method for Analysis of Polarization Resistance Data", *Corrosion*, 37, p. 696, 1981.
21. Barnartt, S., "Tafel Slopes for Iron Corrosion in Acidic Solutions", *Corrosion*, 27, p. 467, 1971.
22. Danielson, M.J., "An Evaluation of the Three-Point Method to Measure Corrosion Rates", *Corrosion*, 38, p. 580, 1982.
23. Jankowski, J. and Juchniewicz, R., "A Four-Point Method for Corrosion Rate Determination", *Corrosion Science*, 20, p. 841, 1980.
24. Bandy, R., "The Simultaneous Determination of Tafel Constants and Corrosion Rate - A New Method", *Corrosion Science*, 20, p. 1017, 1980.
25. McLaughlin, B.D., "A New Approach for Determining Corrosion Current and Tafel Slopes", *Corrosion*, 37, p. 723, 1981.
26. Greene, N.D. and Gandhi, R.H., "Calculation of Corrosion Rates From Polarization Data With A Microcomputer", *Materials Performance*, 21, p. 34, 1982.
27. Devereux, O.F., "Polarization Curve-Fitting by Computer Modelling", *Corrosion*, 35, p. 125, 1979.
28. Devereux, O.F. and Kim, K.Y., "Computer Modelling of Polarization Curves in Passive Systems", *Corrosion*, 36, p. 263, 1980.
29. Taylor, R.J. and Williams, L.F.G., "Corrosion Rate Measurements on Steel in Sulfuric Acid Using a Microprocessor Controlled Potentiostat", *Corrosion*, 36, p. 41, 1980.
30. Forsythe, G.E., Malcolm, M.A. and Moler, C.B., "Computer Methods For Mathematical Computations", Prentice-Hall, Inc., Toronto, p. 76, 1977.
31. *ibid.*, p. 63.
32. *ibid.*, p. 169.
33. *ibid.*, p. 157.

34. *ibid.*, p. 48.
35. Donahue, F.M., "Electrochemical Techniques in Corrosion Studies", Corrosion Chemistry, ACS Symposium Series 89, Brubaker, G.R. and Phipps, P.B.P., Eds., American Chemical Society, Washington, p. 52, 1979.

UNLIMITED DISTRIBUTION

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1 ORIGINATING ACTIVITY Defence Research Establishment Atlantic	2a. DOCUMENT SECURITY CLASSIFICATION UNCLASSIFIED	
	2b. GROUP	
3 DOCUMENT TITLE CORROS: A Computer Program for Analyzing Polarization Resistance Data		
4 DESCRIPTIVE NOTES (Type of report and inclusive dates) Technical Communication		
5 AUTHOR(S) (Last name, first name, middle initial) Hanham, C.M. and Gallagher, P.J.		
6 DOCUMENT DATE APRIL 1986	7a. TOTAL NO. OF PAGES 85	7b. NO. OF REFS 35
8a. PROJECT OR GRANT NO.	9a. ORIGINATOR'S DOCUMENT NUMBER(S) DREA Technical Communication 86/303	
8b. CONTRACT NO.	9b. OTHER DOCUMENT NO.(S) (Any other numbers that may be assigned this document)	
10 DISTRIBUTION STATEMENT Unlimited		
11. SUPPLEMENTARY NOTES	12. SPONSORING ACTIVITY Defence Research Establishment Atlantic	
13. ABSTRACT <p>A program, CORROS, has been written to run on the DEC 20 computer at DREA. It provides a means for analyzing polarization resistance data from potentiodynamic polarization experiments, in order to determine corrosion current densities. CORROS accepts data files which are transferred from an EG&G PARC Model 350 Corrosion Measurement System to the DEC 20 computer with another computer program, <Staal>C11A20¹. A nonlinear least squares curve fitting technique is used to fit a curve, which satisfies the Stern-Geary equation, to the experimental data. The corrosion potential ϕ_{corr}, the anodic and cathodic Tafel constants b_a and b_c, the polarization resistance R_p, and the corrosion current density i_{corr}, along with their estimated errors, and the relative RMS error of the fitted data, are determined and stored in a file LIST. Another file PLOTS, which is accepted by <Staal>SAPLTF² for graphic presentation of the data, is also output.</p>		

DSIN
10-070

KEY WORDS

polarization resistance
corrosion
electrochemical
computer program
PASCAL

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the organization issuing the document.
- 2a. **DOCUMENT SECURITY CLASSIFICATION:** Enter the overall security classification of the document including special warning terms whenever applicable.
- 2b. **GROUP:** Enter security reclassification group number. The three groups are defined in Appendix 'M' of the DRB Security Regulations.
3. **DOCUMENT TITLE:** Enter the complete document title in all capital letters. Titles in all cases should be unclassified. If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital-letter abbreviation in parentheses immediately following the title.
4. **DESCRIPTIVE NOTES:** Enter the category of document, e.g. technical report, technical note or technical letter. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.
5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the document. Enter last name, first name, middle initial. If military, show rank. The name of the principal author is an absolute minimum requirement.
6. **DOCUMENT DATE:** Enter the date (month, year) of Establishment approval for publication of the document.
- 7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.
- 7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the document.
- 8a. **PROJECT OR GRANT NUMBER:** If appropriate, enter the applicable research and development project or grant number under which the document was written.
- 8b. **CONTRACT NUMBER:** If appropriate, enter the applicable number under which the document was written.
- 9a. **ORIGINATOR'S DOCUMENT NUMBER(S):** Enter the official document number by which the document will be identified and controlled by the originating activity. This number must be unique to this document.
- 9b. **OTHER DOCUMENT NUMBER(S):** If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s).
10. **DISTRIBUTION STATEMENT:** Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as:
 - (1) "Qualified requesters may obtain copies of this document from their defence documentation center."
 - (2) "Announcement and dissemination of this document is not authorized without prior approval from originating activity."
11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.
12. **SPONSORING ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring the research and development. Include address.
13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (TS), (S), (C), (R), or (U).

The length of the abstract should be limited to 20 single-spaced standard typewritten lines, 7 1/4 inches long.
14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.

END

Dtic

7-86